Criblage virtuel sur grille de composés isolés au Vietnam

The Quang Bui

N° d'ordre : D. U : 2583
E D S P I C : 701

UNIVERSITE BLAISE PASCAL - CLERMONT II

ECOLE DOCTORALE
SCIENCES POUR L'INGENIEUR DE CLERMONT-FERRAND

T h è s e

Présentée par

QUANG THE BUI

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

SPECIALITE: INFORMATIQUE

Criblage virtuel sur grille de composés isolés au Vietnam

Soutenue publiquement le 26 juin 2015 devant le jury :

M. ou Mme	David Hill	Professeur des universités	Président
	Christian Perez	Directeur de recherches INRIA	Rapporteur
	Ziad El-Bitar	Chargé de recherches CNRS	Rapporteur
	Jean-Marc Pierson	Professeur des universités	Examinateur
	Sorina Camarasu-Pop	Ingénieur de recherches CNRS	Examinateur
	Vincent Breton	Directeur de recherche CNRS	Directeur de thèse
	Emmanuel Medernach	Ingénieur de recherche CNRS	Co-directeur de thèse

TABLE DE MATIERES

LISTE DE FIGURES

LISTE DE TABLE

Remerciements

Je tiens à remercier le Dr. Vincent Breton, Directeur de ma thèse, pour m'avoir encouragée et stimulée dans la réalisation de mes travaux par son exigence, ses apports constructifs, sa vigilance et sa patience.

Je remercie également le Dr. Nguyen Hong Quang, Co-Directeur de ma thèse pour son aide, ses conseils et son support tout au long de cette thèse.

J'adresse tous mes remerciements à Dr. Ziad El Bitar, ainsi qu'à Dr. Christian Perez, de l'honneur qu'ils m'ont fait en acceptant d'être rapporteurs de cette thèse.

Je remercie très sincèrement Dr. Emmanuel Medernach pour la coopération et la participation au jury de thèse.

Je voudrais remercier Dr. Hwang Soonwook et tous mes amis à KISTI qui m'ont fourni des idées et l'environnement de travail pour que je puisse d'enrichir la réalisation de cette thèse.

Je tiens également à remercier Dr. Lydia Maigne, responsable de l'équipe PCSV et tous mes collègues / amis à l'IFI- MSI, IOIT, INPC, l'équipe PCSV et laboratoire LPC, spécialement Doan Trung Tung, Géraldine Fettahi, Sylvia Gervois, Sébastien Cipière et Vu Trong Hieu pour leurs aides et leurs supports.

Enfin, je remercie particulièrement ma famille proche pour son soutien et bien évidemment remercier ma femme NGUYEN Lam Hong qui est toujours à mon côté.

CHAPITRE 1 : INTRODUCTION

1.1 Contexte

L'Institut National de Chimie des Produits (Institute of Natural Products Chemistry ou INPC) de l'Académie des Sciences du Vietnam développe depuis plusieurs années une activité autour de la recherche de nouveaux médicaments issus de la biodiversité au Vietnam. L'INPC a développé une expertise pour extraire des composés chimiques issus de la biodiversité et en caractériser la structure tridimensionnelle. Une criblothèque d'environ 500 composés a ainsi été créée. Le développement d'un nouveau médicament prend de l'ordre d'une dizaine d'années et passe par plusieurs phases. La première de ces phases est la découverte d'un composé avec une action inhibitrice démontrée sur la cible biologique d'une maladie. Par exemple, un composé chimique qui dégrade fortement une fonction biologique du parasite vecteur du paludisme constitue un médicament potentiel pour combattre cette maladie. C'est généralement après la mise en évidence d'une telle action que les composés sont brevetés, en général par les laboratoires ou entreprises qui les ont isolés. Les phases suivantes de développement des médicaments sont prises en charge par les laboratoires pharmaceutiques car elles sont extrêmement coûteuses.

Dans la phase de découverte de nouveaux médicaments, l'activité des composés chimiques sur une cible biologique est mesurée afin de mettre en évidence une action inhibitrice. Le développement d'approches *in silico* pour le criblage virtuel des composés chimiques est une alternative aux approches classiques in vitro beaucoup plus coûteuses à mettre en œuvre. A partir de 2006, dans le cadre de l'initiative WISDOM, l'utilisation des environnements de grilles de calculs pour des déploiements à grande échelle de calculs de criblage virtuel a débouché sur le dépôt de plusieurs brevets pour des molécules actives contre le diabète, le paludisme et le SARS. L'utilisation des grilles informatiques a donc été identifiée comme une voie économiquement prometteuse pour accompagner la recherche de nouveaux médicaments issus de la biodiversité au Vietnam. La stratégie choisie est de répéter le criblage des composés isolés par l'INPC sur une grande variété de cibles biologiques issues de la bibliothèque Protein Data Bank. Celle-ci regroupant plusieurs milliers de structures, il faut

donc envisager de pouvoir soumettre de façon fréquente des projets de criblage virtuel à la grille. La première étape du criblage virtuel est le « docking » dans laquelle un logiciel de chimie computationnelle va calculer la probabilité d'ancrage du composé chimique ou ligand au site actif de la cible biologique. Ce calcul est répété pour chaque couple ligand-cible. Or, l'environnement utilisé dans le cadre de l'initiative WISDOM pour soumettre des centaines de milliers de calculs indépendants de « docking » a montré d'importantes limites. La stratégie de soumission des tâches sur la grille utilisait un modèle très simple dans lequel un opérateur resoumettait manuellement les tâches qui échouaient. Mais celles-ci représentaient une fraction significative des tâches soumises, de l'ordre de 20 à 30%, ce qui entrainait une forte perte d'efficacité et une convergence lente de l'exécution des tâches.

Le développement de nouvelles stratégies basées sur le déploiement d'agents a permis d'améliorer considérablement le taux de succès des tâches et le confort des utilisateurs, ouvrant la voie à une démocratisation de la grille. C'est ainsi que la grille de production française dispose d'un serveur basé sur la technologie DIRAC (Distributed Infrastructure with Remote Agent Control). Baptisé FG-DIRAC, il enregistre les tâches soumises par des utilisateurs issus d'une ou de plusieurs communautés scientifiques et les déploient sur les ressources exposées dans des organisations virtuelles. Cette approche a aussi l'avantage de permettre une migration transparente pour l'utilisateur d'une infrastructure de grille informatique à une infrastructure de nuage (cloud). Une version DIRAC adaptée à l'infrastructure de cloud est en cours de test au Centre de Calculs de l'IN2P3 (CC-IN2P3). En Corée, le KISTI (Korea Institute of Science and Technology Information) a développé la plate-forme HTCaaS (High Throughput Computing as a Service) similaire à DIRAC pour exploiter les ressources de ses clusters et supercalculateurs pour le traitement de données à haut débit, notamment dans perspective de la recherche de nouveaux médicaments.

1.2 Objectifs de la thèse

Dans ce contexte, l'objectif de cette thèse a été d'étudier dans quelle mesure des plates-formes multidisciplinaires telle que FG-DIRAC ou HTCaaS pouvaient répondre aux besoins des chimistes de l'INPC. Il est en effet difficile d'envisager que ceux-ci puissent gérer leur propre serveur alors que l'installation d'un serveur central multidisciplinaire à l'échelle du Vietnam

répond bien au cahier des charges des utilisateurs de la grille et du cloud, à condition bien sûr de disposer d'un accès réseau suffisamment performant.

Dans cette perspective, nous nous sommes intéressés particulièrement au problème posé par le partage équitable d'une plate-forme de soumission de tâches sur la grille par une ou plusieurs communautés d'utilisateurs. Chaque discipline développe ses propres stratégies d'utilisation des ressources fournies par les grilles de calcul : certaines comme la physique des particules privilégient des stratégies centralisées, tandis que d'autres comme la communauté biomédicale sollicitent la grille de façon totalement décentralisée et aléatoire. L'expérience des infrastructures de grille EGEE et EGI en Europe permet de distinguer deux grandes catégories d'utilisateurs : les utilisateurs qui sollicitent les ressources pour un nombre significatif de tâches requérant typiquement de quelques dizaines à quelques centaines d'heures de calcul, et les utilisateurs (« data challenge users ») qui vont lancer des grandes productions nécessitant le traitement de plusieurs milliers de tâches pendant des dizaines, voire des centaines de milliers d'heures de calcul.

Pour caractériser l'expérience de ces utilisateurs et notamment l'équité dans leur traitement, nous avons choisi deux critères : le ralentissement moyen et le ralentissement maximal expérimentés.

En nous appuyant sur les archives d'une infrastructure de grille régionale, nous avons étudié plusieurs stratégies d'ordonnancement des tâches d'une plate-forme soumettant des jobs pilotes en utilisant l'outil de simulation SimGrid. Pour mieux répondre aux besoins des utilisateurs soumettant des tâches très volumineuses, nous avons proposé une nouvelle politique d'ordonnancement.

Nous avons conduit un travail similaire pour une infrastructure de cloud académique.

Nos résultats de simulation ont été confrontés à des résultats expérimentaux obtenus sur un serveur DIRAC dédié installé au LPC déployant des agents pilotes sur la grille EGI et sur un serveur HTCaaS installé sur un serveur au Centre de Calculs National de Corée du Sud (KISTI).

1.3 Structure de la thèse

Le plan du manuscrit est le suivant :

- le chapitre 2 présente un contexte et un état de l'art de l'utilisation des grilles et du cloud en sciences du vivant
- le chapitre 3 présente un état de l'art de l'ordonnancement des plates-formes sur la grille et le cloud
- le chapitre 4 présente la modélisation du problème étudié
- le chapitre 5 présente les performances des politiques d'ordonnancement modélisées avec SimGrid
- le chapitre 6 présente les résultats expérimentaux obtenus avec le serveur DIRAC installé au LPC et avec le serveur HTCaaS au KISTI.
- le chapitre 7 propose des conclusions et des perspectives

1.4 Liste de publications

1.4.1 Conférences internationales (avec comité de lecture)

The Quang BUI, Jik-Soo KIM, Seungwoo RHO, Seoyoung KIM, Sangwan KIM, Soonwook HWANG, Emmanuel MEDERNACH and Vincent BRETON, A Comparative Analysis of Scheduling Mechanisms for Virtual Screening Workflow in a Shared Resource Environment, proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, Guangdong, (China), May 2015

The Quang BUI, Emmanuel MEDERNACH, Vincent BRETON, Hong Quang NGUYEN. *Multi-Level Queue-Based Scheduling for Virtual Screening Application on Pilot-Agent Platforms on Grid/Cloud to Optimize the Stretch*. FUTURE COMPUTING 2014, The Sixth International Conference on Future Computational Technologies and Applications, pp. 32-39, 2014.

The Quang BUI, Emmanuel MEDERNACH, Vincent BRETON, Hong Quang NGUYEN. *Optimizing the Stretch for Virtual Screening Application on Pilot-agent Platforms on*

Grid/Cloud by using Multi-level Queue-based Scheduling. The 4th International Conference on Cloud Computing and Services Science, pp.199-204, 2014

The Quang BUI, Emmanuel MEDERNACH, Vincent BRETON, Hong Quang NGUYEN. *Stretch optimization for virtual screening on multi-user pilot-agent platforms on grid/cloud*. Proceedings of the Fourth Symposium on Information and Communication Technology ACM, pp. 301-310, 2013.

The Quang BUI, Emmanuel MEDERNACH, Vincent BRETON, Hong Quang NGUYEN. *Scheduling of virtual screening application on multi-user pilot-agent platform on grid/cloud to optimize the stretch*. Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics ACM, pp. 692, 2013.

The Quang BUI, Emmanuel MEDERNACH, Vincent BRETON, Hong Quang NGUYEN, et al., *On the Performance Enhancement of the WISDOM Production Environment*. The International Symposium on Grids and Clouds (ISGC), March, 2012, Taipei, Taiwan.

1.4.2 Conférences nationales (avec comité de lecture)

The Quang BUI, Trung Tung DOAN, Van Son NGUYEN, Vincent BRETON, Hong Quang NGUYEN, et al.. *Criblage virtuel sur grille de composés isolés au Vietnam*. Rencontres Scientifiques France Grilles 2011, Sep 2011, Lyon, France.

CHAPITRE 2 : CONTEXTE ET ETAT DE L'ART DE L'UTILISATION DES GRILLES ET DU CLOUD EN SCIENCES DU VIVANT

Dans ce chapitre, nous présentons les acteurs de la recherche de nouveaux médicaments issus de la biodiversité au Vietnam. Ensuite, nous présentons un état de l'art de la recherche de nouveaux médicaments sur la grille. Pour cela, nous présentons les principales infrastructures de grille ainsi que les plates-formes existantes.

2.1 Recherche de nouveaux médicaments issus de la biodiversité au Vietnam

2.1.1 Présentation de la VAST et de l'INPC

L'Académie vietnamienne des sciences et de la technologie (VAST- Vietnam Academy of Science and Technology) est la principale agence gouvernementale scientifique et technologique du Vietnam. Dédiée à l'étude des sciences naturelles et au développement de nouvelles technologies en accord avec les grandes orientations de l'État, son rôle est de fournir une base scientifique et technologique pour la construction de politiques, de stratégies, et de plans de développement socio - économique, ainsi que de former des ressources humaines scientifiques et technologiques de haute qualité pour le pays. La VAST regroupe 30 instituts nationaux et 7 unités non universitaires, 9 entreprises d'État propre, plus de 20

syndicats de production et 35 unités d'instituts, principalement à Hanoi, Ho Chi Minh ville, Haiphong, Nha Trang, Dalat et Hue.

L'Institut de Chimie des Produits Naturels (INPC - Institute of Natural Products Chemistry) de l'Académie Vietnamienne des Sciences et de la Technologie a été créé sur la base du décret 65/CT en date du 5 Mars 1990 par le gouvernement du Vietnam. Les domaines de recherche de l'INPC sont la chimie des produits naturels, la biochimie, la chimie de l'environnement et leurs applications, notamment industrielles. L'INPC s'intéresse notamment à la recherche de composés bioactifs issus de la biodiversité marine et terrestre, dans la perspective de synthétiser des produits valorisables par l'industrie pharmaceutique.

2.1.2 Présentation de la base de données de composés de l'INPC

Au Vietnam, la médecine traditionnelle a une longue histoire de développement. Elle utilise des parties de plantes médicinales naturelles, comme par exemple, les racines, les fleurs, les tiges d'un arbre, les feuilles... Il y a actuellement environ 4000 plantes médicinales enregistrées au Vietnam. L'Institut de Chimie des Produits Naturels de l'Académie des Sciences du Vietnam (INPC) collecte des échantillons issus de la biodiversité locale et détermine la structure tridimensionnelle des molécules isolées. Il y a maintenant environ 200 composés isolés à partir des animaux marins et 300 composés à partir des plantes. L'enjeu pour cet institut est de constituer une base de données des échantillons et de mettre en place une chaîne de traitement des informations structurelles permettant de déterminer sur quelles cibles biologiques les produits isolés sont potentiellement actifs. Comme l'illustre la Figure 2-1, il s'agit d'utiliser des ressources de calcul pour sélectionner parmi les cibles biologiques recensées dans la base de données Protein Data Bank [1] celles sur lesquelles les composés contenus dans la base de données de l'INPC sont les plus prometteurs. Il ne s'agit pas pour l'INPC de développer des médicaments car le coût complet du développement d'un nouveau médicament est typiquement de l'ordre d'un milliard d'euros et requiert l'intervention de laboratoires pharmaceutiques notamment pour toutes les phases d'expérimentation clinique. Par contre, le dépôt d'un brevet international pour une molécule peut intervenir une fois que l'activité inhibitrice de celle-ci sur le virus, la bactérie ou le parasite vecteur d'une maladie est démontrée. Par exemple, les médicaments actuels les plus efficaces dans le traitement du

paludisme utilisent des composés extraits initialement de plantes de la famille de l'artémisine dont l'action était connue dans la médecine traditionnelle chinoise.

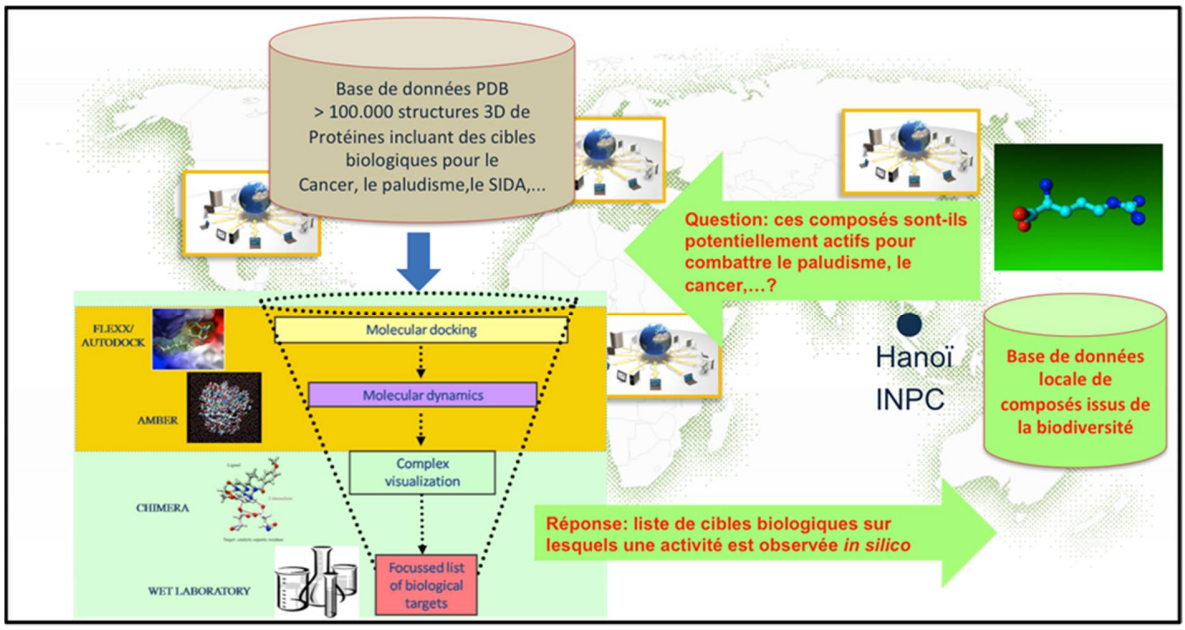


Figure 2-1 : Représentation schématique de la chaine de traitement des composés chimiques isolés par l'INPC

Nous allons maintenant présenter l'état de l'art de l'utilisation des grilles informatiques pour la recherche de nouveaux médicaments.

2.2 Recherche de nouveaux médicaments sur la grille : Etat de l'art

La recherche *in silico* (*in silico* signifie assistée par ordinateur) de nouvelles molécules actives pour combattre une maladie a été identifiée très tôt comme une application prometteuse des grilles de calcul, car la première étape dite de criblage requiert de tester l'action inhibitrice d'un très grand nombre de composés sur une cible donnée [2].

2.2.1 Introduction au criblage virtuel

Le criblage virtuel est la sélection *in silico* des meilleurs candidats médicaments qui agissent sur une protéine cible donnée [3]. Le criblage peut se faire in vitro, à la paillasse, mais son

coût est très élevé : plusieurs euros par composé testé. Multiplié par le nombre de composés ou ligands qui peuvent être synthétisés par les industries chimiques [4], le coût d'un criblage systématique atteint des millions d'euros, somme hors de portée d'un laboratoire de recherche public. Un criblage *in silico* permet de sélectionner un nombre réduit de molécules prometteuses, ramenant le nombre de tests in vitro de quelques millions à quelques centaines.

Le criblage virtuel requiert de connaître la structure tridimensionnelle du site actif de la protéine ciblée, la structure tridimensionnelle du composé chimique ou ligand et un logiciel qui va calculer la probabilité d'ancrage (« docking » en anglais) du ligand sur le site actif de la protéine.

Par exemple, le virus H5N1 de la grippe aviaire porte le nom de deux protéines (H5 et N1) qui sont impliquées dans la réplication des virus et leur propagation de cellule en cellule (Figure 2-2). La neuraminidase N1 est une cible connue des médicaments antiviraux comme le Tamiflu utilisé pour traiter les malades atteints de formes graves de la grippe lors de la pandémie de grippe A de type H1N1 pendant l'été et l'automne 2009.

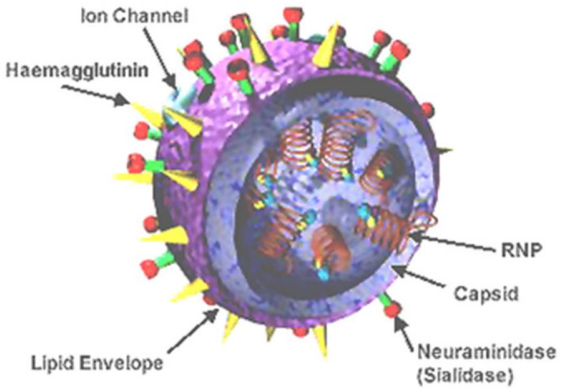


Figure 2-2: Structure du virus influenza

La neuraminidase constitue donc une cible de choix pour la recherche de nouveaux médicaments et plusieurs structures sont documentées dans la Protein Data Bank.

Le criblage virtuel permet la sélection et le classement *in silico* des meilleurs médicaments candidats, c'est à dire les molécules qui pourraient influer sur l'activité biochimique de la cible. Il peut être utilisé comme un filtre pour éliminer les composés toxiques qui sont susceptibles d'échouer dans la phase finale du processus de découverte de médicaments. Le criblage virtuel à haut débit permet ainsi de tester les grandes bibliothèques chimiques où sont enregistrées dans des bases de données les structures de plusieurs millions de composés disponibles dans les laboratoires ou entreprises.

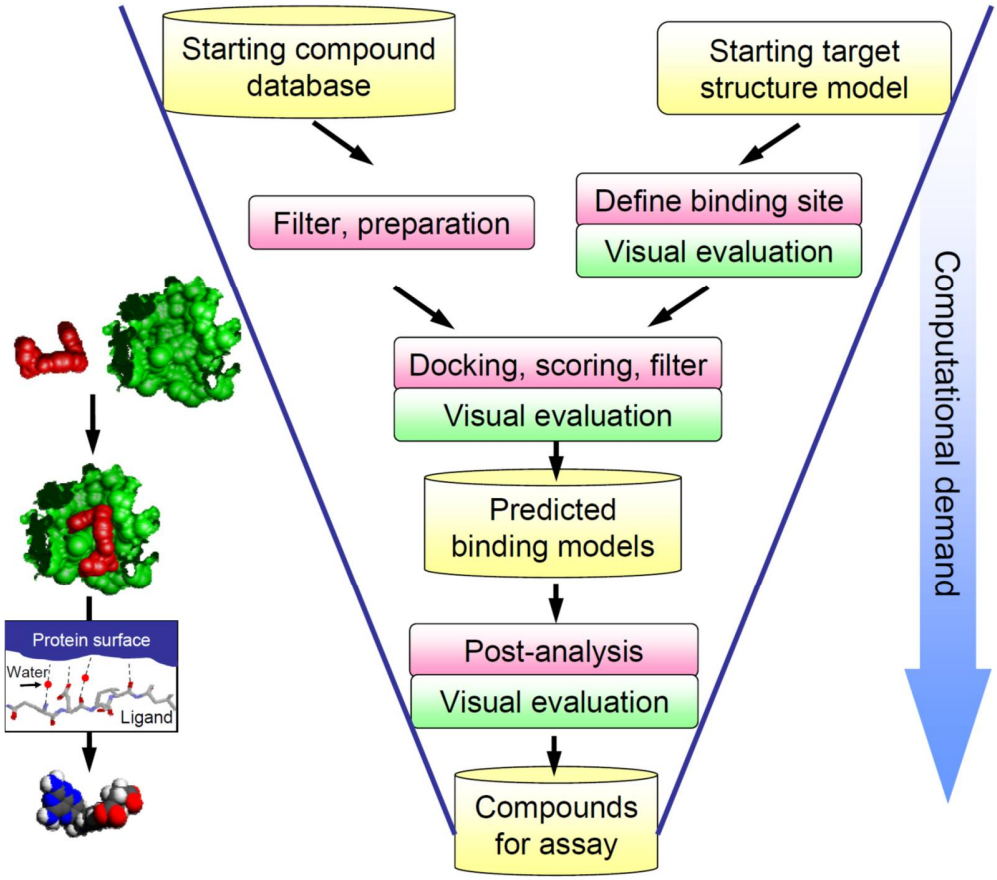


Figure 2-3: Etapes du criblage virtuel à haut débit

La Figure 2-3 décrit les étapes typiques d'un projet de criblage virtuel d'une base de données de composés sur une cible spécifique. Sa mise en œuvre requiert des points de contrôle visuel et éventuellement des cycles automatiques répétés.

La première étape consiste à choisir la cible biologique dans la Protein Data Bank et à préparer la base de données de composés chimiques ou ligands. Des bibliothèques de structures tridimensionnelles de composés sont disponibles sur internet, soit qu'elles soient issues de la recherche publique et fournies par des laboratoires universitaires, soit qu'elles soient proposées par les industriels qui commercialisent les composés. Il est possible aussi d'utiliser des bases de données privées de ligands comme celle de l'INPC.

Il est possible de cibler la recherche sur des familles de composés préalablement identifiés ou d'avoir une approche très globalisée en testant de façon aléatoire des millions de ligands. Cette deuxième approche est beaucoup plus coûteuse en temps de calcul.

Après le choix des ligands, il est nécessaire de préparer la description de la structure tridimensionnelle du site actif de la protéine cible, en ajoutant notamment des atomes d'hydrogène et des molécules d'eau.

L'étape suivante dite de « docking » est le calcul de l'énergie de liaison du ligand au site actif en utilisant un algorithme d'évaluation (scoring algorithm) par les modèles et les approximations. Beaucoup de logiciels de docking sont disponibles (Table 2-1) : ils diffèrent notamment par les modèles et approximations utilisés pour le calcul de l'énergie de liaison.

Nom	Editeur	Site Internet
Autodock	Scripps	http://www.scripps.edu/mb/olson/doc/autodock/
Surflex	Biopharmics	http://www.biopharmics.com/products.html
FlexX	BioSolveIT	http://www.biosolveit.de/FlexX/
Dock	UCSF	http://dock.compbio.ucsf.edu/
Glide	Schrödinger	http://www.schrodinger.com/Products/glide.html
Gold	CCDC	http://www.ccdc.cam.ac.uk/products/life_sciences/gold/
LigandFit	Accelrys	http://www.accelrys.com/cerius2/c2ligandfit.html
Fred	OpenEyes	http://www.eyesopen.com/products/applications/fred.html
ICM	Molsoft	http://www.molsoft.com/products.html

Table 2-1: Logiciels de « docking »

La post-analyse, la dernière étape, est alors nécessaire avant l'inspection visuelle finale. La notation de consensus [5], l'incorporation d'énergies de solvatation, une meilleure description des interactions électrostatiques [6] et l'analyse géométrique de l'aire de surface composé-protéine peuvent être utilisés pour cette fin.

La dynamique moléculaire [7] permet un traitement souple des complexes formés par la cible et le ligand à température ambiante et est en mesure d'affiner les orientations des ligands afin de trouver des complexes plus stables. Elle permet également la reclassement des molécules sur la base de fonctions d'évaluation plus précise [8].

La plupart des logiciels de docking, notamment les plus populaires comme Autodock, ne requièrent pas de configuration particulière en termes de mémoire vive. A l'inverse, les logiciels de dynamique moléculaire sont très gourmands en mémoire vive et tournent sur des supercalculateurs. Le docking a donc été rapidement identifié comme une application particulièrement adaptée au calcul distribué, notamment sur grille de calcul.

2.2.2 Introduction à la grille de calcul

La grille de calcul consiste en un ensemble de ressources informatiques réparties géographiquement qui sont utilisées pour atteindre un but commun. La grille est donc un

système distribué. Les grilles sont souvent construites à l'aide de bibliothèques de logiciels à usage général appelées communément intergiciels en français ou middleware en anglais. Voici une liste des quelques termes importants qui seront répétés tout au long de cette thèse :

Computing Element (CE) : L'élément de calcul est un service qui donne accès à un gestionnaire de ressources local. Sa fonction principale est la gestion des travaux ou jobs (défini ci-après), par exemple : la soumission des jobs, le contrôle des jobs, etc… Le CE peut être utilisé directement par un utilisateur ou à travers le système de gestion des charges ou Workload Management System (WMS, défini ci-après également) qui soumet un job à un CE approprié en fonction des spécificités de ce job.

Certificat et proxy : Afin de s'authentifier auprès des ressources de la grille, un utilisateur doit avoir un certificat numérique délivré par une autorité de certification et reconnu par une Organisation Virtuelle (VO, définie ci-après également). Le certificat utilisateur, dont la clé privée est protégée par un mot de passe, est utilisé pour générer et signer un certificat temporaire, appelé proxy, qui est utilisé pour l'authentification auprès des services de la grille. Parce qu'un proxy est une preuve d'identité, le fichier qui le contient doit être lisible uniquement par l'utilisateur, et un proxy a, par défaut, une durée de vie courte (généralement 12 heures) pour des raisons de sécurité.

Job : Un job est une unité de travail, un programme avec des paramètres et des entrées. Soumis par un utilisateur, il s'exécute sur les ressources de la grille pour produire des résultats de sorties. Une application de grille peut se résumer à un seul job, ou requérir l'exécution de plusieurs jobs similaires (c'est-à-dire un même programme qui est exécuté plusieurs fois avec des paramètres ou entrées différents, comme dans le cas du docking) ou impliquer un enchaînement (flot) de jobs différents qui s'exécutent sur la grille.

Agent-pilote : Un agent-pilote ou job pilote est un job générique soumis par un utilisateur sur la grille dont le rôle est de réserver une ressource de la grille qui sera utilisée ultérieurement par un programme réel. Un job pilote peut être utilisé pour lancer plusieurs tâches (programmes).

Job Description Language (JDL) : le langage de description de job permet de décrire un job avant de le soumettre. Il permet de préciser par exemple l'exécutable à lancer et ses paramètres, les fichiers à déplacer vers et à partir du WN (Worker Node, défini ci-après) sur

lequel le job est exécuté, les fichiers d'entrée nécessaires sur la grille, et toutes les autres exigences du CE et du WN.

Storage Element (SE): un élément de stockage fournit un accès uniforme à des ressources de stockage de données. L'élément de stockage peut contrôler des serveurs de disques simples, des piles de disques de grande taille ou d'autres moyens de stockage. Le SE peut prendre en charge différents protocoles d'accès aux données.

Tâche : une tâche est une unité de travail, un programme avec des paramètres et des entrées qui s'exécutent sur la grille. Dans cette thèse, nous différencions les tâches des jobs de la façon suivante : une tâche n'est pas soumise directement par l'utilisateur mais est lancée par un job pilote. Une tâche est donc un travail qu'un job pilote peut exécuter au cours de son existence sur la grille.

Worker Node (WN) : c'est la machine où les jobs sont exécutés. Un Computing Element s'appuie sur un ensemble de WNs pour exécuter les jobs.

Workload Management System (WMS) : Le but du système de gestion de charge est d'accepter des jobs utilisateurs, de les assigner au CE le plus approprié, d'enregistrer leur statut et de récupérer leur résultat. L'utilisateur peut soumettre un job à un CE directement mais en général, il laisse le WMS choisir le CE le plus approprié à partir de la description de ce job.

Virtual Organisation (VO) : une organisation virtuelle se réfère à un ensemble dynamique d'individus ou d'institutions définis autour d'un ensemble de règles et de conditions de partage de ressources. L'appartenance à une VO accorde des privilèges spécifiques à un utilisateur dans l'accès aux ressources. Pour devenir membre d'une VO, un utilisateur doit se conformer aux règles de l'utilisation de la VO.

Virtual Organisation Membership Service (VOMS) : VOMS est un système de gestion des données d'autorisation d'accès aux ressources de la grille. VOMS fournit une base de données de rôles et de fonctions d'utilisateur et un ensemble d'outils pour l'accès et la manipulation. VOMS utilise le contenu de cette base de données pour générer des certificats de grille pour les utilisateurs lorsque cela est nécessaire.

2.2.3 Les grilles pour les sciences du vivant en Europe

Les sciences du vivant sont un très vaste domaine scientifique regroupant de nombreuses disciplines. Parmi celles-ci, la bioinformatique, la biologie structurale et l'imagerie médicale ont été pionnières dans l'utilisation des grilles. La bioinformatique constitue l'ensemble des méthodes informatiques pour gérer, organiser et analyser les données biologiques. La biologie structurale est la branche de la biologie qui étudie la structure et l'organisation spatiale des macromolécules biologiques. Un de ses champs d'application privilégiés est la recherche de nouveaux médicaments *in silico* comme nous l'avons vu dans les paragraphes précédents. L'imagerie médicale regroupe les moyens d'acquisition et de restitution d'images du corps humain à partir de différents phénomènes.

Dès le démarrage des projets européens d'infrastructure de grilles, bioinformatique et imagerie médicale ont constitué des domaines d'application [9] privilégiés. Mais l'adoption des grilles à grande échelle dans ces champs disciplinaires s'est heurtée à plusieurs obstacles. Les besoins spécifiques de l'imagerie médicale en matière de sécurité et de la bioinformatique en termes de flexibilité et de gestion des données n'ont pas été satisfaits par les intergiciels de grille déployés sur les infrastructures européennes et l'utilisation de la grille reste encore limitée par rapport aux besoins en croissance exponentielle. Malgré cela, les sciences du vivant constituent la deuxième plus grosse communauté d'utilisateurs de l'infrastructure européenne EGI comme nous allons le voir plus en détails dans le paragraphe suivant.

L'INFRASTRUCTURE DE GRILLE EUROPEENNE EGI

EGI (European Grid Initiative) est une infrastructure européenne distribuée qui a pris la suite des projets européens DataGrid et EGEE (Enabling Grids for E-SciencE). Les pays partenaires d'EGI participent à sa coordination par le biais d'Initiatives de Grilles Nationales (National Grid Initiatives ou NGIs en anglais), organisations mises en place pour coordonner et animer les efforts dans l'opération et l'utilisation de ressources informatiques de grille et de cloud distribuées au niveau national. Elles constituent des points de contact et des structures d'animation pour les communautés de recherche et les centres de ressources. L'Initiative de Grille Européenne EGI fédère les efforts des Initiatives de Grilles Nationales pour fournir une

infrastructure intégrée au niveau européen et ouverte à toutes les disciplines. Les principaux services offerts par EGI à travers les NGIs sont les suivants:

- Services d´opération et de sécurité : opération d'une infrastructure de calcul et de données dans le monde entier, fiable et hautement disponible 24 heures sur 24, 7 jours sur 7
- Services aux utilisateurs: support aux utilisateurs, formations, etc.
- Services de liaisons : connexion avec les autres infrastructures non-européennes

Le soutien de la Commission Européenne à EGI se matérialise par un co-financement du projet EGI-InSPIRE (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe), effort de collaboration impliquant plus de 50 institutions dans plus de 40 pays débuté le 1er mai 2010 et prolongé jusqu'au 31 Décembre 2014. Son objectif est de mettre en place une infrastructure de grille européenne durable (EGI - European Grid Infrastructure). Selon le rapport annuel de EGI pour l'année 2013[1], le nombre total de cœurs fournis par EGI a atteint 347.307, tandis que le nombre total de ressources de calcul des infrastructures intégrées dans EGI est de 373.235, en augmentation de 30,7% depuis l'année précédente. La capacité totale de stockage est de 177 Petaoctets, en augmentation de 25,36% depuis l'an dernier.

La gestion des ressources de l'infrastructure de EGI est faite par plusieurs intergiciels de grille. Les intergiciels constituent le lien qui tient ensemble les éléments de la grille. Ils fournissent des services généraux pour l'ensemble de l'infrastructure : gestion de la charge, gestion de données, authentification, surveillance, etc. Les intergiciels déployés sur EGI doivent respecter le cahier des charges de l' « Unified Middleware Distribution » (UMD). Les quatre intergiciels aujourd'hui déployés sur les infrastructures d'EGI sont : gLite [10], UNICORE [11], Globus Toolkit [12] et ARC [13].

[1] Annual Report 2013 : : https://documents.egi.eu/public/ShowDocument?docid=1664

Discipline	May 12–April 13		May 11 – April 12		Jobs	CPU wall time
	% CPU n. wall time (A)	% of Jobs done (B)	% CPU n. wall time (C)	% of jobs done (D)	(yearly increase from May 11) (E)	(yearly increase from May 11) (F)
High-Energy Physics	93.78	89.58	93.60	91.58	**+1.22%**	**+40.97%**
Infrastructure	0.10	2.88	0.20	3.26	-8.70%	-29,67%
Life Sciences	1.52	4.34	1.30	1.75	**+156.79%**	**+65.12%**
Astrophysics	2.82	1.82	2.25	1.58	**+18.57%**	**+76.64%**
Multidisciplinary	0.12	0.17	0.39	0.48	-62.77%	-56.97%
Others Disciplines	0.59	0.45	1.23	0.72	-36.713%	-32.12%
Unknown Discipline	0.43	0.27	0.20	0.29	**-3.08%**	**+199.45%**
Comput. Chemistry	0.48	0.22	0.38	0.03	**+83.04%**	**+78.31%**
Fusion	0.01	0.10	0.37	0.13	-24.56%	-96,98%
Earth Sciences	0.15	0.11	0.10	0.05	**+139.95%**	**+123.45%**
CS and Mathematics	0.00	0.07	0.00	0.03	+170.56%	-68.06%

Table 2-2: L'usage des ressources par discipline

Le Table 2-2 présente l'usage annuel des ressources par disciplines de Mai 2011 à Avril 2013. Les sciences du vivant représentent 1.52% de l'usage de CPU, en troisième position après la physique des hautes énergies et l'astrophysique. Le nombre de jobs soumis représente 4.34% du nombre total de jobs sur la grille EGI, en deuxième position après la physique des hautes énergies.

En termes de nombre d'utilisateurs, la communauté des sciences du vivant est la plus représentée après la physique des particules au niveau national (Figure 2-4).

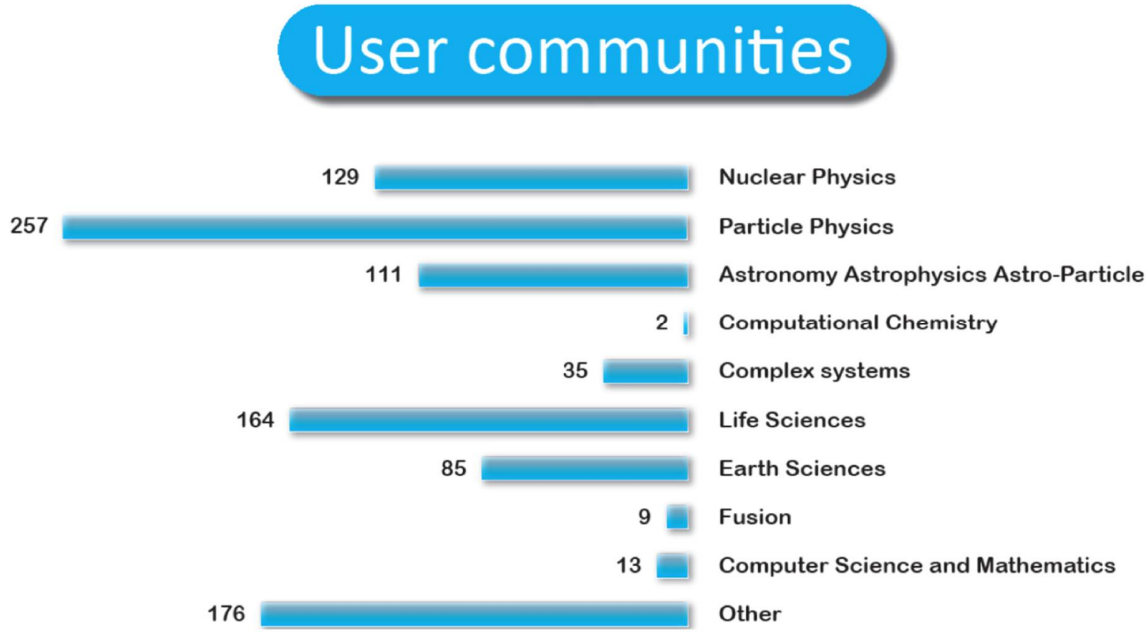


Figure 2-4: Distribution thématique des utilisateurs titulaires d'un certificat émis par France Grilles

ORGANISATION DES SCIENCES DU VIVANT SUR LA GRILLE EGI

Les Organisations virtuelles (Virtual Organization ou VO) sont des groupes de chercheurs ayant des intérêts et des exigences scientifiques similaires, qui souhaitent travailler en collaboration et/ou partager des ressources (par exemple, les données, les logiciels, l'expertise, CPU, espace de stockage), indépendamment de leur emplacement géographique. Les chercheurs doivent adhérer à une VO afin d'utiliser les ressources de l'infrastructure EGI. Chaque VO est autorisée à accéder à un sous–ensemble des ressources d'EGI. Le choix des VOs autorisées à accéder à un site donné de la grille relève de l'entière compétence et autorité de l'administrateur du site. Tout utilisateur muni d´un certificat d´authentification délivré par une autorité de certification acceptée au niveau international peut demander à rejoindre une VO. Chaque organisation virtuelle gère sa propre liste de membres, selon les exigences et les objectifs de la VO. EGI fournit les services centraux permettant aux VOs de se déployer sur des ressources internationales et de tirer le meilleur parti de ces ressources.

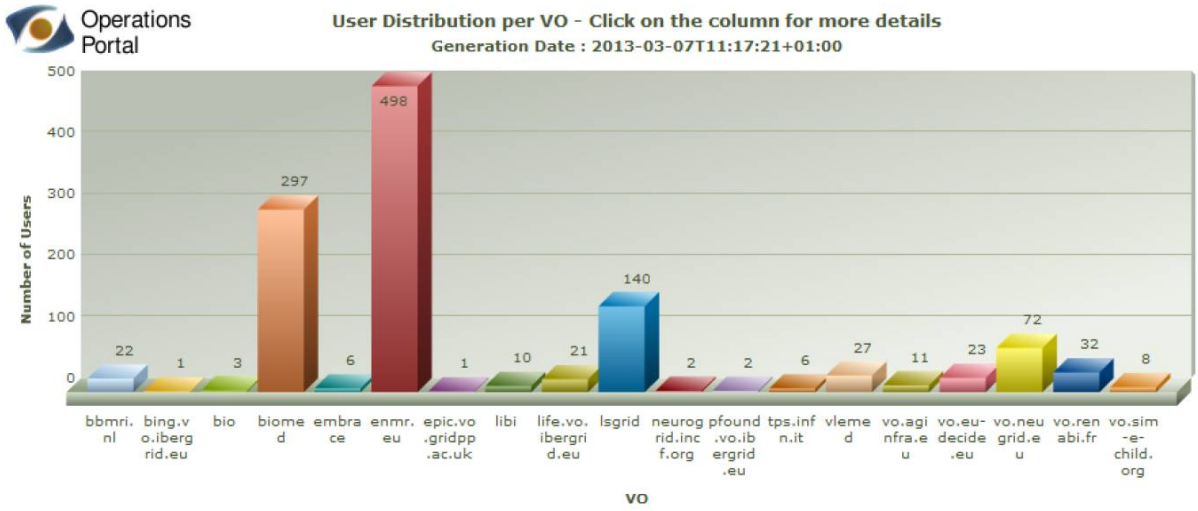


Figure 2-5: Distribution de nombre total d'utilisateurs par organisation virtuelle (VO)

Plusieurs organisations virtuelles (VOs) sont aujourd'hui déployées sur EGI, représentant des communautés d'utilisateurs de tailles variées comme l'illustre la Figure 2-5 qui montre la distribution du nombre total d'utilisateurs par VO du secteur des sciences du vivant. La plus grande par le nombre d'utilisateurs enregistrés est la VO enmr.eu, dédiée à la biologie structurale et plus spécifiquement à la communauté utilisant les techniques de résonance magnétique nucléaire (Nuclear Magnetic Resonance).

Une partie des travaux décrits dans cette thèse a été réalisée sur la VO Biomed qui comptait au 1er Janvier 2014 329 utilisateurs de 20 pays différents. Cette VO a vocation à couvrir de façon très large tous les domaines liés aux sciences du vivant mais la plupart des utilisateurs sont issus de l'imagerie médicale, la bioinformatique et la biologie structurale. Cette VO est librement accessible à des universitaires et à des sociétés privées à des fins non commerciales. La VO Biomed donne accès à des dizaines de milliers de cœurs dans de nombreux sites à travers l'Europe et le monde entier. Ces sites ne sont pour la plupart pas des laboratoires de la discipline.

Face à la multiplication des VOs et pour faciliter la coordination des efforts dans chaque discipline, les Communautés de Recherche Virtuelles (Virtual Research Community - VRC) constituent un mécanisme pour représenter les intérêts d'un domaine de recherche au sein de l'écosystème d'EGI. Ces VRCs peuvent inclure une ou plusieurs organisations virtuelles et ont vocation à être le principal canal de communication entre les chercheurs et EGI. EGI établit des partenariats avec les VRC par le biais d'une convention (Memorandum of Understanding - MoU). Après le processus d'accréditation et l'accord final, les représentants

des VRCs deviennent les interlocuteurs d'EGI dans leur domaine scientifique et profitent des nombreux avantages d'un partenariat solide avec EGI. Ils représentent leur discipline pour négocier des ressources, assurer la liaison avec les Initiatives de Grille Nationales et d'autres fournisseurs de ressources à travers le monde. EGI propose des workshops et des forums pour aider et supporter les communautés sur les problèmes techniques spécifiques, et afin aussi de les impliquer dans l'évolution de l'infrastructure de production d'EGI. Les représentants des VRCs expriment à EGI le cahier des charges de leurs exigences techniques et de service qui est pris en compte dans le développement global de l'infrastructure. Vis-à-vis des communautés qu'ils représentent, ils servent de point de contact pour les nouveaux utilisateurs et apportent de l'aide pour partager les expertises, éviter la réplication des efforts et encourager le partage des ressources, des données et des outils. Ils organisent des événements de formation et fournissent des services pour opérer et supporter les VOs communes, opérer des services partagés, etc. Ils contribuent enfin à diffuser les informations et les connaissances afin de faciliter la communication interne et externe avec d'autres groupes d'intérêt.

La Communauté Virtuelle de Recherche des Sciences de la Vie (Life Science Grid Community - LSGC) rassemble les domaines scientifiques suivants: la bioinformatique, la biologie moléculaire, les bio-banques, l'imagerie médicale, la biologie structurale et les neurosciences.

En résumé, EGI fournit une plate-forme de calcul et de stockage de données pour les communautés scientifiques en Europe. EGI est un environnement où les utilisateurs peuvent se rencontrer et travailler ensemble dans le cadre de collaborations internationales. Aujourd'hui EGI est une infrastructure de ressources en expansion et elle continue à évoluer pour inclure de nouveaux types de ressources et notamment des « nuages » ou « clouds » académiques.

La grille en France

France Grilles[2] est l'Initiative de Grille Nationale (National Grid Initiative) Française établie depuis Juin 2010 sous la forme d'un Groupement d'Intérêt Scientifique entre 8 organismes majeurs de recherche :

[2] Site web de Frace Grilles : http://www.france-grilles.fr/

- Le Ministère de l'Education Nationale, de l'Enseignement Supérieur et de la Recherche (MENESR)
- Le CNRS
- Le Commissariat à l'Energie Atomique et aux énergies alternatives (CEA)
- L'Institut National de la Recherche Agronomique (INRA)
- L'Institut National de Recherche en Informatique et en Automatique (INRIA)
- L'Institut National de la Santé et de la Recherche Médicale (INSERM)
- La Conférence des Présidents d'Université (CPU)
- Le REseau NAtional de télécommunications pour la Technologie, l'Enseignement et la Recherche (RENATER)

Les missions de France Grilles sont les suivantes :

- Établir et opérer une infrastructure nationale de grille de production, pour le traitement et le stockage de grandes masses de données scientifiques.
- Contribuer avec les autres états membres impliqués au fonctionnement de l'infrastructure européenne EGI.
- Favoriser les rapprochements et les échanges entre les équipes travaillant sur les grilles de production et les grilles de recherche.

Ces missions s'étendent aujourd'hui au domaine du « cloud computing ».

L'infrastructure de production France Grilles intègre plus de 32.000 processeurs et 31 Petaoctets de stockage répartis dans 18 sites à travers le territoire français pour les analyses de données à haut débit.

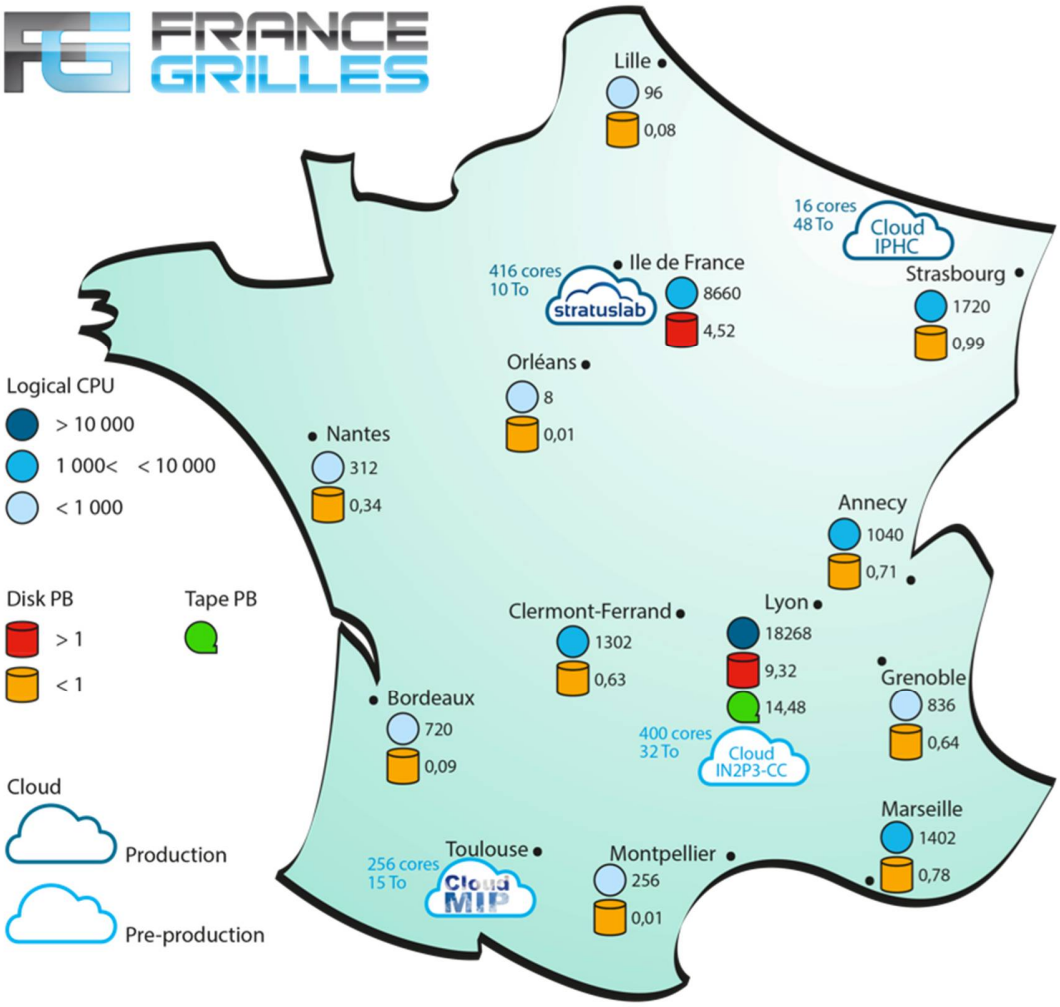


Figure 2-6: Carte des sites fournissant des ressources de grille et de cloud dans France Grilles

Ces ressources sont mises à la disposition des utilisateurs français – il y a aujourd'hui environ 850 titulaires de certificats de grille en France - mais aussi des chercheurs à travers le monde via des organisations virtuelles internationales. France Grilles fait partie des plus gros contributeurs à EGI (Figure 2-7), avec l'Allemagne, l'Italie et le Royaume-Uni.

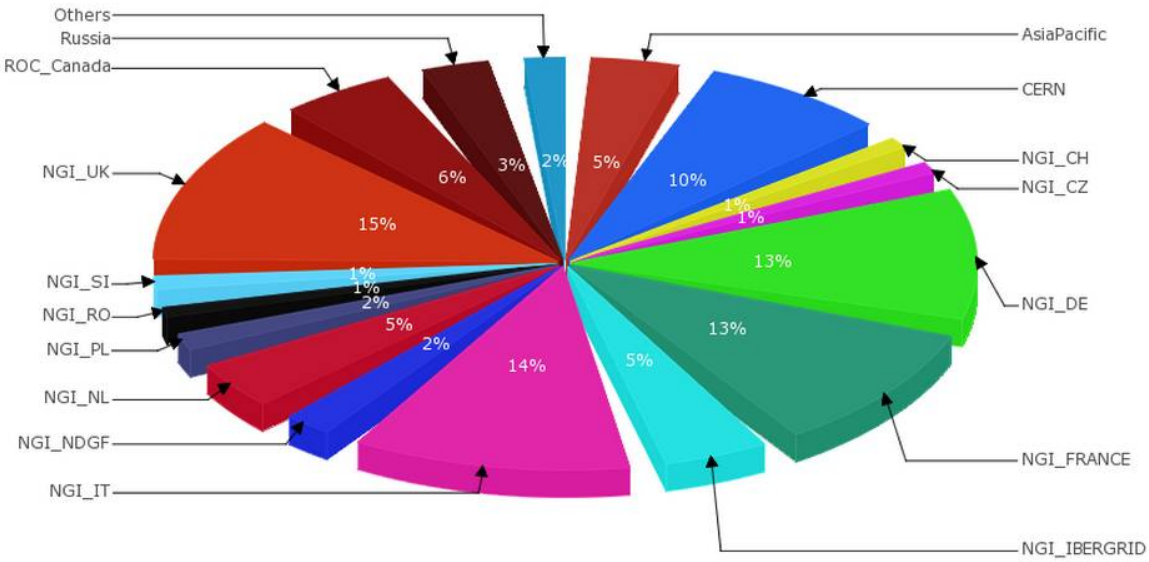


Figure 2-7: Temps de calcul normalisé fourni pendant l'année 2013 par les différentes initiatives de grille nationales

Les communautés de recherche les plus actives en France sur la grille sont les suivantes:

- L'astronomie, l'astrophysique et les astroparticules, principalement pour la simulation Monte-Carlo de nouveaux détecteurs comme CTA (Cerenkov Telescope Array), l'analyse des données issues d'instruments comme AUGER, GLAST ou LSST dans le futur, ou pour la modélisation de systèmes astrophysiques complexes
- La physique des particules, plus gros consommateur de ressources pour l'analyse des données du LHC mais aussi pour d'autres grandes expériences internationales, comme l'expérience D0 à FERMILAB
- Les sciences de la Planète, notamment la sismologie, les sciences de l'atmosphère, l'hydrologie, l'hydrodynamique, l'exploration géophysique et l'étude du climat
- Les sciences du Vivant aujourd'hui très actives sur la grille notamment dans les domaines suivants :
    - la bioinformatique autour du Réseau National des plateformes de Bioinformatique (RENABI), avec des infrastructures dédiées comme Décrypthon.
    - l'imagerie médicale, au sein de la Life Science Grid Community[3]
    - les neurosciences, à travers des projets nationaux et européens

[3] http://www.lsgc.org

Des communautés plus réduites sont très actives dans le domaine des systèmes complexes ou de la chimie tandis que l'observatoire de la grille se propose de collecter et d'analyser les données sur le comportement de la grille[4] pour les chercheurs en informatique. La croissance très rapide des besoins de traitement des données dans de nombreux domaines scientifiques amène régulièrement de nouvelles communautés à frapper à la porte de la grille.

RENABI/GRISBI

ReNaBi (Réseau National des plaques-formes Bioinformatiques - http://www.renabi.fr/) est un réseau de bioinformatique qui a pour objectif de favoriser la coordination de l'activité des nombreuses plates-formes des différents instituts pour mieux répondre aux besoins des équipes de recherche en biologie à l'échelle nationale. Le réseau ReNaBi rassemble 28 plates-formes au sein des centres régionaux suivants: APLIBIO (Paris – Île de France), PRABI (Rhône-Alpes), ReNaBi-GO (Grand Ouest), ReNaBi-GS (Grand Sud), ReNaBi-NE (Nord-Est), ReNaBi-SO (Sud Ouest). Ces plates-formes offrent de nombreuses ressources bioinformatiques pour les communautés scientifiques en sciences de la vie.

Au sein du réseau ReNaBi, l'initiative GRISBI (Grid Support to Bioinformatics) a été lancée pour rassembler les ressources de six plates-formes bioinformatiques françaises dans une grille dédiée pour rendre possibles des applications bioinformatiques à grande échelle: génomique comparative et annotation de génome, biologie des systèmes, prédiction de la fonction des protéines, interaction moléculaire comme protéine-protéine ou d'ADN-protéines. Les six centres de base partagent et mutualisent leurs ressources de stockage et de calcul, mais aussi des ressources comme des bases de données et des logiciels, comme illustré sur la Figure 2-8.

[4] http://www.grid-observatory.org

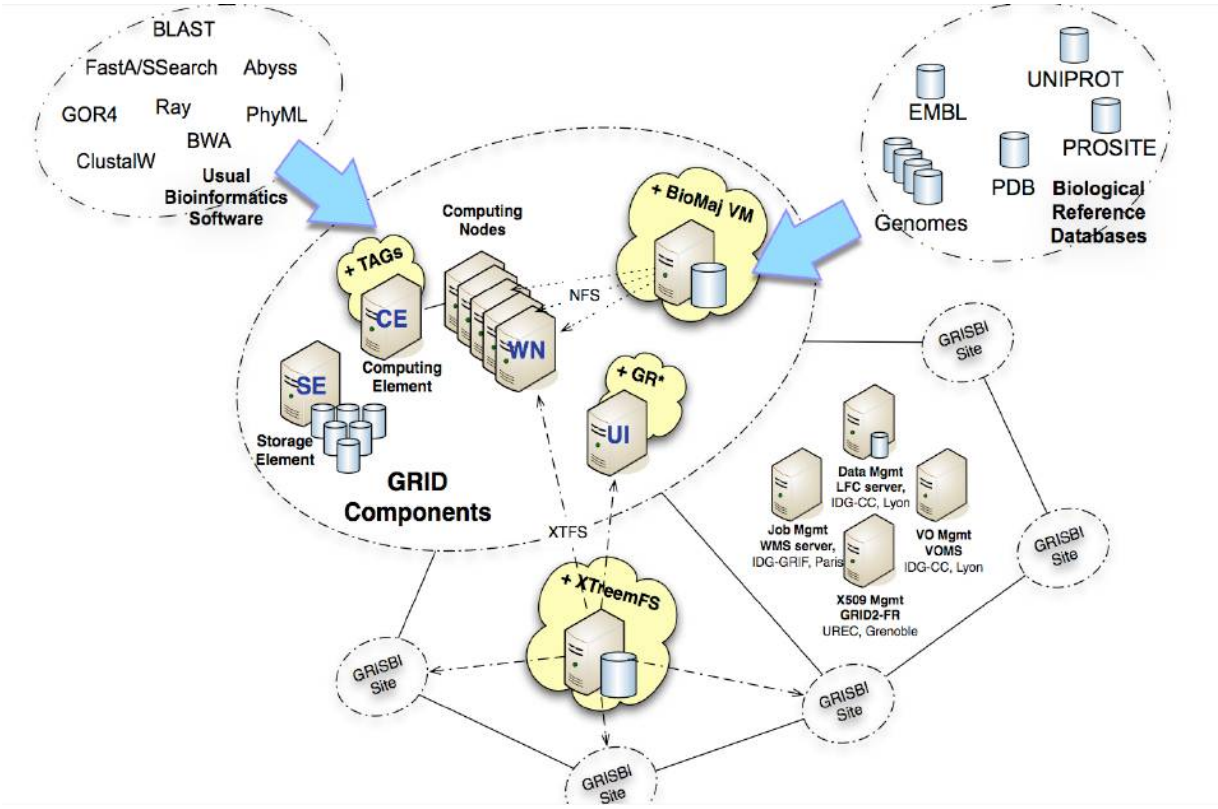


Figure 2-8: Infrastructure GRISBI (Source : http://www.isima.fr/~mephu/FILES/JOBIM12/jobim_actes_2012_clef.pdf)

Cette figure présente l'infrastructure GRISBI avec les composants de gLite standard (en gris) et les composants supplémentaires (en jaune): BioMAJ, XtreemFS [14] et GR * commandes. Différents composants de la grille sont fournis par le middleware gLite [10]: User Interface (UI), Computing Element (CE), Worker Node (WN), Storage Element (SE), Workload Manager System (WMS), Logic File Catalog (LFC), Berkeley Database Information Index (BDII). La plate-forme GRISBI utilise les systèmes de stockage des fichiers de gLite et de XtreemFS [14]. L'outil BioMAJ (http://biomaj.genouest.org/) déployé sur GRISBI permet de synchroniser et de mettre à jour les données sur chaque site. Des logiciels particulièrement utilisés par la communauté de bioinformatique comme ClustalW [15], FastA [16], PhyML [17], sont pré-installés sur les nœuds de la grille GRISBI gère quelques banques de données biologiques : UNIPROT (http://www.uniprot.org/), UNIREF [18] et la Brookhaven Protein Data Base [1]. Pour les services centraux de la grille, GRISBI collabore avec France Grilles en utilisant ses services.

Décrypthon

Fruit d'une collaboration entre le CNRS, l'Association Française contre les Myopathies (AFM) et la société IBM, le programme Décrypthon (http://www.decrypthon.fr) a fourni à partir de 2004 aux équipes de recherche en bioinformatique des ressources de calcul et de stockage pour aider la recherche sur les maladies neuromusculaires et les maladies rares. Il a utilisé pour cela des ressources de calcul distribuées installées par IBM dans six universités françaises (Bordeaux 1, Lille 1, Paris 6 Jussieu, ENS Lyon, Crihan à Rouen et Orsay) et/ou des ordinateurs personnels via le projet World Community Grid (http://www.worldcommunitygrid.org/index.jsp) . Ce projet propose aux internautes de mettre leur ordinateur personnel à la disposition des chercheurs en téléchargeant un logiciel qui permet d'intégrer l'ordinateur à une grille pendant le temps où il est disponible.
Ce sont ainsi des centaines de milliers d'ordinateurs qui ont pu être mobilisés à travers le monde sur des projets scientifiques et notamment la recherche de nouveaux médicaments pour les maladies négligées.

2.2.4 Etat de l'art de la grille au Vietnam

Les acteurs

Le calcul haute performance au Vietnam se concentre dans des centres de calcul dispersés à travers le pays. Ces centres de taille modeste (de quelques dizaines à quelques centaines de cœurs) sont installés dans les universités, les instituts et les organisations et fonctionnent de manière indépendante, pratiquement sans coordination, ni partage des ressources et des services.

On peut citer le Centre de Calcul de Haute Performance de l'Université Polytechnique de Hanoi (HPC-HUST), le Centre de Prévision Hydro-météorologique Nationale, les Centres de Calculs de l'Université des Sciences Naturelles de l'Université Nationale de Hanoi, de l'Université Polytechnique de Hanoï et de l'Université Nationale de Ho Chi Minh-Ville. L'Académie des Sciences et Technologies du Vietnam (VAST) héberge aussi des ressources informatiques à l'Institut de Mathématiques, l'Institut des Technologie de l'Informatique et l'Institut Militaire des Sciences et Techniques.

L'Institut de la Francophonie pour l'Informatique (IFI) est un institut international en informatique créé par l'Agence universitaire de la Francophonie (AUF) en 1995 pour répondre à une demande vietnamienne en formation de cadres supérieurs en informatique et pour aider à l'émergence de professeurs d'informatique vietnamiens de calibre international au niveau universitaire.

Pour le Vietnam, l'IFI représente une aide au développement et à la formation de formateurs et une passerelle vers l'acquisition, l'utilisation et le développement local des techniques avancées d'information et de communication. C'est dans ce contexte que l'IFI s'est intéressé à partir de 2007 à la technologie des grilles de calculs.

Histoire de la grille au vietnam

La première initiative de grille au Vietnam est venue des Etats-Unis à travers le projet PRAGMA (Pacific Rim Application and Grid Middleware Assembly - http://www.pragma-grid.net/) en 2002. Son objectif était d´établir des collaborations et de promouvoir l'utilisation des technologies de grilles entre les communautés de chercheurs des grandes institutions à travers le Pacifique.

La collaboration entre la France et le Vietnam sur la grille a démarré à l'automne 2007 où une première école, baptisée ACGRID, a été organisée dans les locaux de l'Institut des Technologies de l'Informatique (Institute Of Information Technology - IOIT) de l'Académie des Sciences et Technologies du Vietnam (VAST) dans le cadre de la série des écoles Do-Son. Les objectifs de cette école étaient très ambitieux :

- former des administrateurs systèmes vietnamiens à la gestion des services de grille
- démarrer 5 sites de la grille EGEE au Vietnam grâce à du matériel acheté par le CNRS pour la formation
- former des utilisateurs de ces sites pour démarrer le déploiement d'applications scientifiques

L'école n'a pas atteint tous ses objectifs mais 3 sites à Hanoï (Centre de Calcul de Haute Performance de l'Université Polytechnique de Hanoi, Institut de la Francophonie pour l'Informatique et Institut des Technologies de l'Information de la VAST) ont pu effectivement rejoindre la grille dans les mois suivants.

Mais rapidement, l'installation des sites s'est heurtée à des problèmes d'infrastructure réseau. En effet, toutes les connexions entre les sites académiques au Vietnam et en Europe utilisaient le réseau TEIN2 (http://tein2.archive.dante.net/server/show/nav.621.html), projet international financé par la Commission Européenne avec les objectifs suivants:

- Augmenter la connectivité à Internet pour la collaboration dans les domaines de la recherche et l'éducation entre l'Europe et l'Asie
- Améliorer la connectivité intra-régionale en Asie
- Agir comme un catalyseur pour le développement des réseaux de recherche nationaux dans les pays en développement dans la région Asie-Pacifique.

Bien que l'objectif de TEIN2 ait été d'offrir une infrastructure de connexion à haute vitesse qui satisfasse aux conditions requises pour des sites de grille, en pratique, le processus de déploiement des nœuds de la grille au Vietnam s'est heurté à de multiples problèmes d'accès à TEIN2, d'instabilité de la connexion, de capacités limitées de gestion,…

Dans ce contexte difficile, le projet d'Initiative de Grille Nationale VNGrid a été lancé en 2008 avec pour objectif de connecter et de partager les ressources entre les organisations participantes. A cause des difficultés rencontrées, notamment au niveau de l'infrastructure réseau, ce projet n'a pas réussi à poser les fondations d'une infrastructure nationale exploitable et durable. Il a par contre produit des résultats académiques et quelques publications dans le domaine de la recherche et du développement sur cluster de calcul et sur grille.

Malgré ces difficultés, en 2009, l'Institut de la Francophonie pour l'Informatique (IFI) a eu l´opportunité de participer au projet européen EuAsiaGrid (http://www.euasiagrid.org/) et est devenu un des instituts les plus actifs et engagés dans des activités de recherche sur la grille et le cloud. L'IFI a accueilli les écoles ACGRID en 2009 et 2012 et joué un rôle moteur dans l'animation de l'activité scientifique sur la grille et le cloud depuis lors.

Le portail g-INFO de surveillance de la grippe aviaire [19] a été développé dans le cadre du projet EuAsiaGrid par Dr. Doan Trung Tung pendant son travail de thèse. g-INFO est une solution complète pour aider les utilisateurs non experts de la grille à analyser et construire les arbres phylogénétiques sur les séquences de virus influenza. Un deuxième outil de déploiement de logiciel bioinformatique sur la grille est e-Panam, une solution pour le traitement de données métagénomiques issues de séquençage haut débit [19].

D´autres projets de recherche et de développement basés sur la grille et le cloud sont en cours entre l´IFI et les partenaires de la VAST comme le projet sur le criblage virtuel des composants naturels issus de la biodiversité avec l´Institut de Chimie des Produits Naturels (INPC), objet de cette thèse ou le projet entre l'IFI et l´Institut Physique du Globe (IGP) et l´Institut des Technologie de l'information (IOIT) sur la modélisation d'une base de scénarios de tremblement de terre pour le Centre d'Études, de Surveillance et d'Alerte de risque de tsunami dans la mer de l´Est.

2.3 Des grilles au «cloud computing»

2.3.1 Introduction

Bien qu'il y ait un certain flou sur la définition précise, la plupart s'accordent pour définir le Cloud Computing ou informatique en nuage comme un système où les ressources sont exposées comme des services [20]. L'idée de fournir les technologies de l'information comme un service (IT-as-a-service) offre des avantages énormes aux clients qui veulent une puissance de calcul importante sans devoir acheter des ordinateurs coûteux. La capacité à augmenter ou à diminuer la puissance de calcul à la demande en payant seulement la puissance consommée est particulièrement intéressante pour la gestion des pics de charge.

Le cloud computing s'appuie sur l'utilisation d'environnements virtualisés d'exécution (virtualized running environment). De cette façon, via le découplage du matériel, le fournisseur des ressources n'a plus à supporter de multiples exigences des systèmes d'exploitation tandis que le développeur ne dépend plus de la plate-forme du fournisseur, comme c'était le cas pour les grilles. L'utilisateur a donc la possibilité de configurer des environnements de fonctionnement dynamiques sans la médiation du fournisseur de ressources. Un environnement d'exécution encapsulée permet d'éviter une défaillance dans le système d'exploitation des machines virtuelles et de les affecter sur le même hôte.

La technologie de l'informatique en nuage est le résultat d'un processus évolutif qui a commencé il y a 20 ans et dont la grille de calcul a constitué certainement une étape. Elle se situe à la convergence des grilles, de la virtualisation et de l'automatisation. Bien que la

fiabilité et la sécurité des clouds pose encore question, de nombreuses entreprises ont pris les premières mesures en vue de son adoption, soit en utilisant un Software-as-a-Service (SaaS), soit en louant des cycles de calcul à des fournisseurs. Une autre approche dans les grands groupes industriels est le déploiement de clouds internes privés.

2.3.2 Clouds pour les sciences du vivant en France

Le monde académique s'est emparé de cette évolution technologique en parallèle des grands acteurs industriels du secteur. Plusieurs piles logicielles pour l'administration de clouds font l'objet d'un développement collaboratif open source. On peut citer notamment Openstack (Open Source Cloud Computing Software: http://www.openstack.org), OpenNebula (Open Source Data Center Virtualization: http://www.opennebula.org) et StratusLab (http://stratuslab.eu).

France Grilles développe une infrastructure nationale de type IAAS (Infrastructure As A Service) construite sur la base d'une fédération de ressources de calcul et de stockage distribuées sur un ensemble de sites français [21]. Comme illustré sur la Figure 2-6, quatre sites sont aujourd'hui intégrés à cette fédération, à Lyon, Paris, Strasbourg et Toulouse. Trois autres sites sont en cours d'installation à Grenoble, Lille et Montpellier.

Bien que physiquement distribuées, les ressources de cette infrastructure doivent pouvoir être utilisées de manière homogène et transparente. Elles doivent aussi pouvoir faire partie intégrante d'infrastructures plus larges, telles que celle proposée dans le cadre du projet européen EGI-Inspire (INtegrated Sustainable Pan-european Infrastructure for Researchers in Europe). Ce projet coordonne le déploiement d'une offre européenne de type IaaS de calcul et de stockage pour la communauté scientifique. Baptisée EGI FedCloud (EGI Federated Cloud -), cette infrastructure de production est l'agrégation de multiples fournisseurs de ressources Cloud utilisant différents gestionnaires. La fédération de clouds France Grilles doit intégrer cette fédération EGI Fed Cloud dans les prochains mois.

L'utilisation des infrastructures de cloud dans les sciences du vivant n'en est qu'à ses débuts. Quelques groupes ont expérimenté l'utilisation des clouds publics : on peut citer par exemple l'expérience de déploiement du pipeline EOULSAN de traitement de données de séquençage haut débit à travers les solutions Amazon Web Services [22] qui sont flexibles et économiques mais peuvent poser des problèmes en terme de sécurité et de temps de transfert

des données. Les développeurs d'Eoulsan ont déployé l'outil sur la grille EGI avec des performances observées significativement supérieures à celles d'Amazon.

Dans le domaine académique, nous allons présenter plus en détails deux initiatives importantes de mise à disposition de ressources cloud pour les communautés des sciences du vivant : le projet e-biothon [23] et le projet IFB-core. Ces deux projets ont en commun le déploiement d'une ressource cloud à l'IDRIS (Institut du Développement et des Ressources en Informatique Scientifique), un des trois centres de calcul hébergeant des supercalculateurs d'envergure nationale (Tier-1).

E-Biothon

IBM, le CNRS (via l'IDRIS et l'Institut des Grilles et du Cloud avec le soutien de France Grilles), l'InstitutFrançais de Bioinformatique, l'INRIA et SysFera se sont associés pour mettre à disposition des chercheurs la plate-forme E-Biothon qui consiste en deux racks de Blue Gene/P. Ces deux racks offrent une puissance en crête de 28 téraflops et chaque rack compte mille vingt-quatre noeuds, de quatre cœurs chacun. Chaque nœud a une quantité de mémoire RAM partagée de 2 gigaoctets. La Blue Gene est aussi dotée d'une capacité de stockage de 200 téraoctets. Pour accéder à cette puissance de calcul, un serveur dédié est utilisé comme machine frontale. À travers ce portail développé par la société SYSFERA [24], les chercheurs ont accès à tout un environnement de travail leur permettant d'exécuter simplement les traitements informatiques en lien avec les analyses « omiques » à réaliser, puis de gérer les données générées, tout cela à partir d'un simple navigateur web, sans installation locale et avec une sécurité des données garantie. Ainsi, ils peuvent interagir avec une seule interface conviviale plutôt qu'avec des gestionnaires de ressources de Calcul Haute Performance. Le portail s'occupe ainsi d'abstraire et de rendre transparente l'infrastructure de calcul aux utilisateurs afin de leur permettre de se concentrer sur leurs recherches.

La plate-forme E-Biothon est utilisée depuis quelques mois par trois équipes qui y déploient des analyses de phylogénie, d'épidémiologie et de génomique.

Institut français de bioinformatique

L'institut Français de Bioinformatique est une infrastructure de service en bio-informatique issue d¹un appel à propositions « Infrastructures en Biologie et Santé » du programme national des «Investissements d'Avenir».

Ce projet implique des plates-formes de bioinformatique dépendant des cinq principaux organismes publics de recherche, CNRS, INRA, INRIA, CEA et INSERM, des universités et des Instituts Pasteur et Curie. Au total, ce sont une vingtaine de plates-formes bioinformatiques regroupées en six pôles régionaux couvrant le territoire national qui sont ainsi impliquées dans l'IFB.

L'IFB a pour mission principale de fournir les services de base en bioinformatique à la communauté des sciences de la vie dans les domaines de la génomique, de la protéomique, etc. L'IFB est également le nœud français d'ELIXIR, une initiative européenne, menée par l'EBI (European Bioinformatics Institute) qui vise à fournir, au niveau européen, des services similaires à ceux que l'IFB fournit au niveau français.

Au cœur des infrastructures de l'IFB, un cloud académique est en cours de déploiement à l'IDRIS. L'ambition est qu'il fournisse 10.000 cœurs et 1 PetaOctet de stockage pour la gestion et l'analyse des données de sciences du vivant, en complément des 6000 cœurs et 1 PetaOctet de stockage fournis par les plates-formes distribuées sur le territoire national. Après le choix de la pile logicielle pour la gestion du cloud qui s'est porté sur StratusLab, les premiers tests ont démarré au printemps 2014.

2.3.3 Les clouds au Vietnam

Le Vietnam est un marché à forte potentialité pour le cloud computing avec un nombre important d'entreprises qui ont décidé d'utiliser ce type de services [25]. Les organisations au Vietnam adoptent les services de cloud computing pour réduire les coûts, pour accéder à des applications liées à la gouvernance d'entreprise et pour la capacité de test à distance et de développement des logiciels. L'intérêt des entreprises vietnamiennes pour le cloud computing a grandi du fait du volume croissant d'informations auxquelles elles sont confrontées tous les jours. En outre, il y a une tendance croissante au Vietnam de permettre aux employés de travailler à distance (télétravail) avec leurs appareils mobiles personnels. VMWare a conjecturé que le pourcentage d'entreprises au Vietnam utilisant le cloud computing serait plus élevé que les autres pays d'Asie du Sud-Est. Un rapport indique que certains experts estiment que le marché du cloud computing au Vietnam va quadrupler d'ici à 2015 [26]. Cette technologie promet donc d'être utilisée largement dans les entreprises et les instituts de recherche au Vietnam à l'avenir.

2.4 Plates-formes sur grille et cloud pour la recherche de nouveaux médicaments

2.4.1 Introduction

Nous venons de voir dans les paragraphes précédents que des infrastructures distribuées pour le stockage et l'analyse des données scientifiques étaient aujourd'hui disponibles et utilisées par les équipes de recherche en sciences du vivant. Aujourd'hui, les grilles de calcul ont des performances remarquables en termes de fiabilité et de disponibilité tandis que les clouds académiques sont dans une phase de montée en puissance semblable à celle connue par les grilles au début des années 2000. A cette époque, les grilles présentaient des limitations importantes en termes de fiabilité et performances et pour améliorer l'expérience des utilisateurs, l'utilisation de plates-formes s'est développée.

Comme nous l'avons vu aussi précédemment, l'étape de criblage virtuel dans la recherche de nouveaux médicaments *in silico* peut être efficacement accélérée sur la grille en déployant sur un très grand nombre de nœuds les calculs d'ancrage (« docking ») entre le site actif de la cible biologique et les composés chimiques ou ligands. Le calcul de l'énergie de liaison entre le ligand et le site actif utilise des logiciels de chimie computationnelle dont le plus connu est Autodock [27] et requiert typiquement quelques minutes par docking sur un PC traditionnel. Le nombre de ligands recensés dans les chimiothèques pouvant atteindre des millions, un projet de criblage virtuel peut requérir des dizaines de millions de calculs si plusieurs cibles biologiques ou plusieurs configurations des cibles, voire plusieurs jeux de paramètres des logiciels de docking sont testés.

L'utilisation de plates-formes s'est donc imposée naturellement pour les projets de criblage virtuel sur grille. Nous allons présenter de façon détaillée trois plates-formes développées en Europe et en Asie : WISDOM, GVSS et HTCaaS.

2.4.2 WISDOM

La plate-forme WISDOM [28], développée par le Laboratoire de Physique Corpusculaire (LPC) Clermont-Ferrand, France, a été utilisée avec succès dans les projets WISDOM-I (2005) et WISDOM-II (2006) comme une couche entre les utilisateurs et les ressources de la grille. Elle facilite la soumission massive des jobs de docking sur la grille. Baptisée WPE pour WISDOM Production Environment, elle est construite au-dessus de l'intergiciel gLite. L'approche adoptée par la collaboration WISDOM fut de développer un système permettant de centraliser la gestion de plusieurs milliers de tâches soumises simultanément sur la grille.

La **Error! Not a valid bookmark self-reference.** montre l'architecture de WPE. Ce système fut conçu au départ sur une stratégie dite « PULL » : Tout d'abord, le client archive ses fichiers de docking dans un fichier compressé et le sauvegarde sur un élément de stockage (SE) dans la grille. Ensuite, il enregistre la métadonnée de sa tâche de docking dans le Task Manager (TM). Le module Job Manager (JM) soumet des jobs pilotes sur la grille. Les jobs pilotes envoient des requêtes de tâche au Task Manager : en d'autres termes, le job pilote récupère sa tâche à partir du Task Manager. S'il y a des tâches dans la file d'attente du Task Manager, il fournit la métadonnée de tâche au job pilote. Le job pilote télécharge le contenu de la tâche et l'exécute. Lorsque les jobs finissent avec succès, les données sont collectées sur des éléments de stockage de la grille. Si les jobs échouent pour une raison connue ou inconnue, ils sont resoumis. Le module WIS (WISDOM Information System) met à jour constamment l'état des jobs pilotes sur la grille. Cette approche pragmatique a permis de faire le premier déploiement à très grande échelle de calculs et a constitué ainsi le premier déploiement d'un « data challenge » dans un autre domaine autre que la physique des hautes énergies sur la grille EGEE.

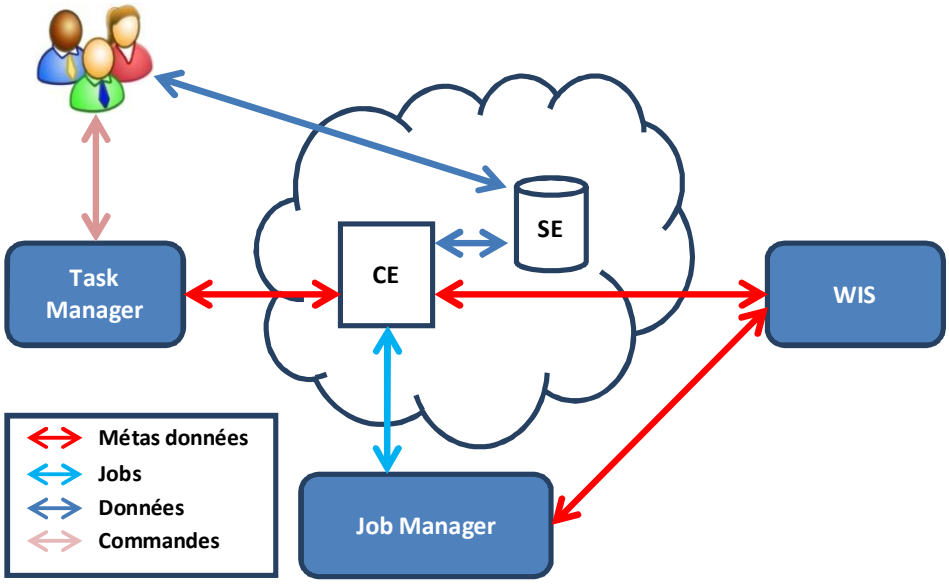


Figure 2-9: L'architecture de WPE

Des déploiements à grande échelle se sont succédés de 2005 à 2011 en ciblant des maladies différentes.

En 2005, le premier déploiement à grande échelle sur l'infrastructure de la grille européenne EGEE, baptisé data challenge WISDOM-I, a utilisé avec succès plus de 80 années de temps CPU en 6 semaines pour une première expérience « *in silico* » [28]–[30]. Il s'agissait de tester environ un million de composés chimiques (médicaments potentiels) avec deux logiciels de docking sur 5 protéines de la famille des plasmepsines, cible biologique potentielle pour combattre la malaria. Cette première étape a permis de sélectionner environ 100.000 composés qui ont fait l'objet d'une deuxième étape de tri [31] par une analyse de dynamique moléculaire. Celle-ci a permis de réduire à 100 le nombre de composés qui ont fait l'objet d'une expérimentation biologique in vitro et finalement in vivo.

En 2006, au moment où les premiers cas de grippe aviaire étaient confirmés en France, une collaboration internationale [32] étudiait l'impact de certaines mutations de la neuraminidase N1 sur l'efficacité des traitements de la grippe H5N1 sur plusieurs milliers de processeurs appartenant à trois infrastructures de grille différentes (Auvergrid-France, EGEE -Europe et TWGrid-Taiwan). Ces études montrèrent que certaines mutations impactaient très significativement l'efficacité du Tamiflu. Elles permirent aussi de tester 300.000 composés chimiques et d'en sélectionner environ 2000 dont les plus prometteurs ont fait l'objet de tests in vitro en Corée du Sud.

En 2007, le data challenge WISDOM-II [33] mobilisait environ 4 siècles de temps CPU pour le criblage virtuel de quatre autres cibles biologiques d'intérêt pour la lutte contre le paludisme, élargissant la recherche à des cibles biologiques du parasite Plasmodium vivax, vecteur d'une forme atténuée de paludisme.

En parallèle de l'analyse des résultats obtenus au cours de ces différentes campagnes de calcul, l'activité dans le cadre de la collaboration WISDOM s'est développée en Corée du Sud où plusieurs centaines de milliers de composés chimiques ont été testés sur des cibles biologiques du SARS [34] et du diabète en 2010 et 2011 [35].

2.4.3 GVSS

Développé à l'Academia Sinica Grid Computing Center (ASGC) de Taïwan, acteur majeur de la grille en Asie, GVSS-1 (Grid-enabled Virtual Screening Services 1 - http://gap.grid.sinica.edu.tw/) est une plate-forme dédiée à la biologie structurale et plus spécifiquement au « docking » pour la recherche de nouveaux médicaments [36]. La plate-forme GVSS-1 offre à l'utilisateur la possibilité de personnaliser la préparation de la base de données ligand et donne accès à plus de 700 protéines de la PDB pour soumettre la simulation de docking. Après la simulation, l'utilisateur peut visualiser les meilleures conformations et des résultats tels que l'histogramme d'énergie et l'analyse en composante principale 2D/3D.

GVSS-1 est basé sur la plate-forme GAP (Grid Application Platform) et utilise l'outil de déploiement DIANE [32] qui déploie des jobs pilotes sur la grille. Les données biochimiques sont stockées en utilisant le catalogue de métadonnées AMGA [37]. L'architecture du système de GVSS est illustrée dans la figure 2-10.

Grâce à DIANE, GVSS-1 peut diviser et exécuter les jobs soumis en plusieurs sous-tâches indépendantes. L'exécution de ces sous-tâches indépendantes tire profit de la disponibilité de la ressource de la grille. GVSS-1 a été utilisé pour la recherche de médicaments *in silico* sur les maladies négligées et émergentes [36].

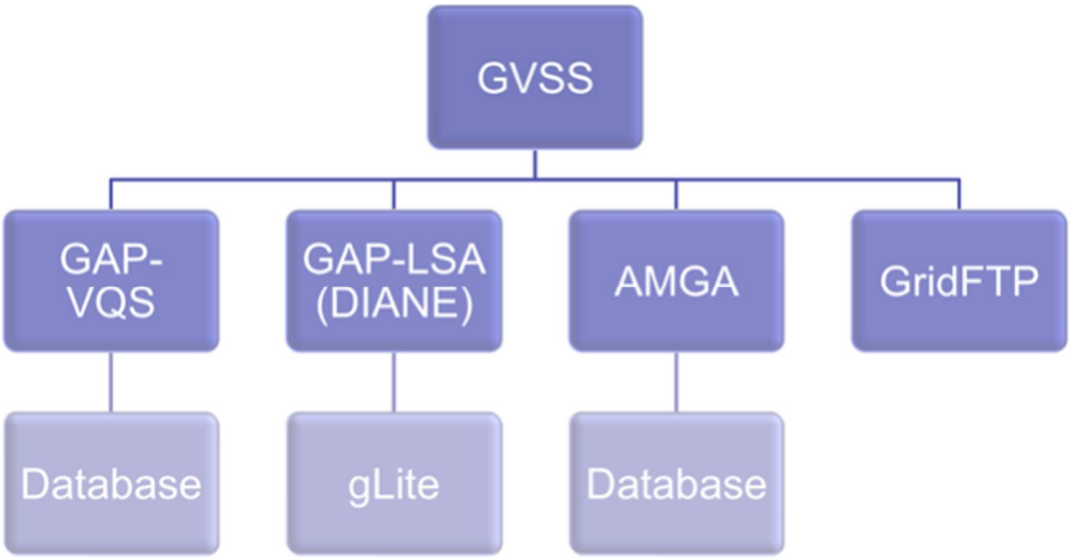


Figure 2-10: Structure de GVSS-1

Après le succès de GVSS-1, ASGC a développé GVSS-2 basé sur la technologie Web [38], [39]. Au lieu de déployer et d'exécuter l'application sur l'ordinateur, ASGC a déployé le système sur un portail web. Un utilisateur muni de son certificat peut participer à la VO EUASIA et peut ainsi accéder à ce portail. Il peut gérer les différentes étapes de préparation du ligand et de la protéine, de déploiement du docking et de traitement des résultats sur le portail, ainsi que de visualisation de ces résultats. Et tout cela se fait à travers une interface web conviviale et facile à utiliser.

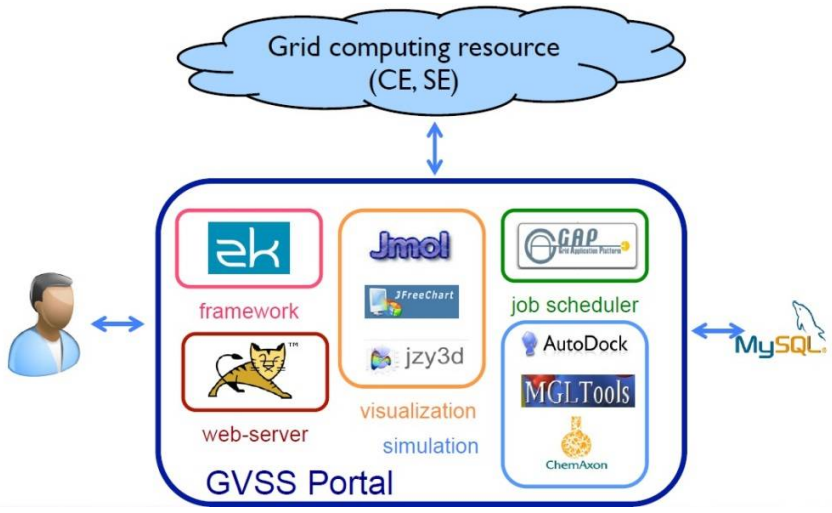


Figure 2-11: Structure du portail GVSS

La Figure 2-11 présente la structure de GVSS-2. Il est basé sur des modules de portail existants (ZK Framework,Tomcat web-server). Les modules de criblage virtuel intègrent des outils de simulation (Autodock, MGL Tools, ChemAxon) et de visualisation (Jmol, jzy3d, JfreeChart).

Bien qu'il soit encore dans un processus de test à petite échelle, GVSS-2 fournit déjà une interface conviviale pour les utilisateurs non spécialistes de l'informatique.

2.4.4 HTCaaS

HTCaaS (High Throughput Computing as a Service) est une plate-forme récemment développée au KISTI pour supporter le déploiement d'applications scientifiques dans le domaine de la pharmacie et de la physique des hautes énergies sur l'infrastructure nationale de calcul intensif en Corée [40]. Elle vise à fournir aux chercheurs des outils pour une exploration complexe et à grande échelle des problèmes scientifiques. HTCaaS permet aux utilisateurs de soumettre efficacement un grand nombre de jobs à la fois par la gestion efficace et l'exploitation de toutes les ressources de calcul disponibles.

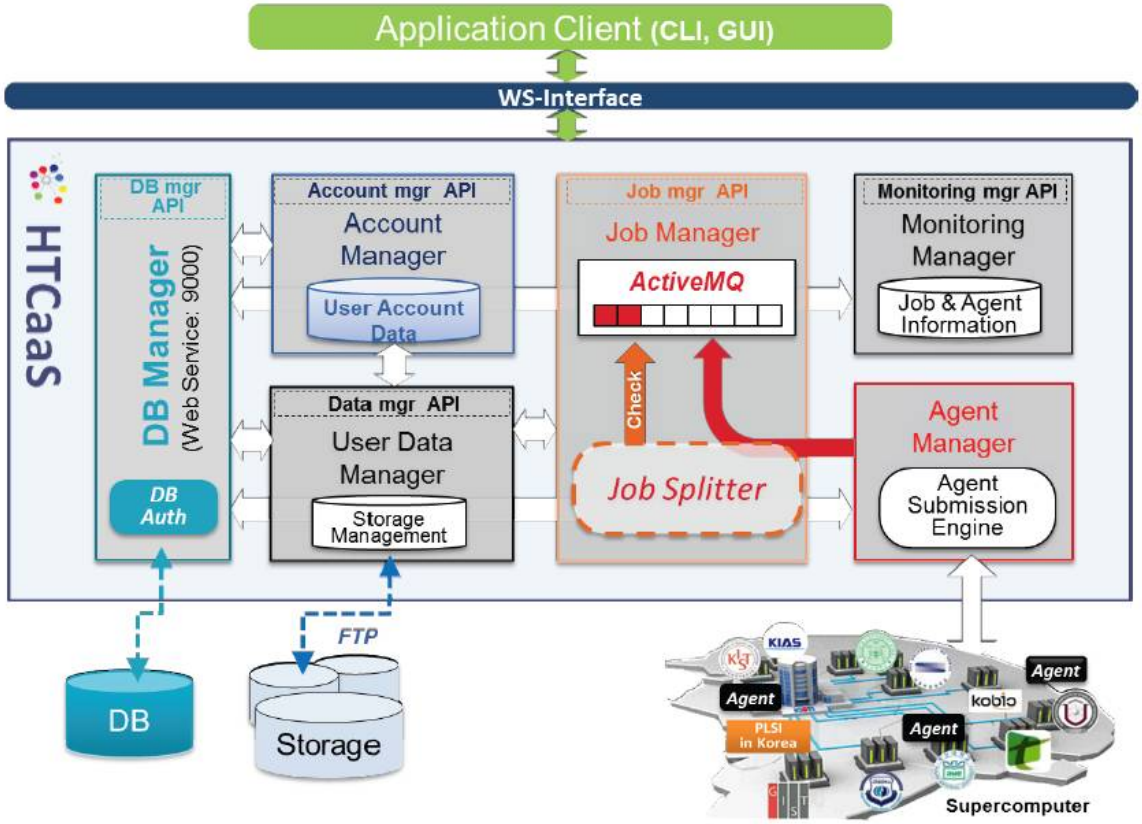


Figure 2-12: La structure de HTCaaS (Source : HTCaaS: Leveraging Distributed Supercomputing Infrastructures for Large-Scale Scientific Computing [41])

La Figure 2-12 montre l'architecture générale du système HTCaaS. Le gestionnaire de comptes (Account Manager) gère les informations de l'utilisateur et fournit des mécanismes d'authentification et d'autorisation intégrés pour accéder à diverses infrastructures informatiques. Le gestionnaire des données des utilisateurs (User Data Manager) est responsable de la gestion des données de l'utilisateur pendant l'exécution des jobs. Le gestionnaire de tâches (Job Manager) effectue l'essentiel de la gestion de jobs. Il maintient les files d'attente séparées par utilisateur, reçoit une métadescription des jobs ou « Meta-job » (écrit en JSDL [42]) qui peut être composée de plusieurs tâches d'un utilisateur, valide le « Meta-job », divise automatiquement le «Meta-job» en de multiples tâches, et contrôle l'exécution de chaque tâche. Le « Monitoring Manager » vérifie périodiquement les exécutions de jobs et des agents actifs en interagissant avec « DB Manager », et si nécessaire, il déclenche des mécanismes de récupération d'échec pour les agents. Une fois que les jobs sont soumis dans le système HTCaaS, les agents (mis en œuvre en Java) sont soumis à partir de l'« Agent Manager » et exécute les tâches sur les ressources de calcul distribuées géographiquement. Une fois déployé, chaque agent appelle activement les tâches, les exécute

et enregistre les statistiques de façon indépendante sur le « DB Manager » qui peut être utilisé pour la surveillance des tâches et des agents par le « Monitoring Manager ».

HTCaaS maintient des files d'attente et gère un pool d’agents propre à chaque utilisateur. Chaque agent tire activement les tâches de sa file d'attente qui correspond à un utilisateur spécifique, et s'il n'y a pas plus de tâches à traiter, il libère automatiquement les ressources et se termine.

2.4.5 DIRAC

DIRAC est défini par ses créateurs [43] comme un intergiciel. La plate-forme DIRAC facilite le calcul scientifique en exposant les ressources de calcul distribué d’une manière transparente aux utilisateurs. Par rapport aux autres plates-formes décrites dans ce chapitre, DIRAC n’a pas été utilisé pour des déploiements à grande échelle de calculs de criblage virtuel. Il est par contre très utilisé en imagerie médicale à travers la plate-forme VIP et plus généralement à travers le serveur DIRAC-FG opéré par France Grilles pour servir de nombreuses communautés d’utilisateurs.

Développée initialement au CPPM à Marseille puis dans la collaboration LHCb au CERN pour satisfaire les besoins liés à la génération d'un grand volume de données de simulation Monte-Carlo, la plate-forme DIRAC (Distributed Infrastructure with Remote Agent Control) est une solution complète pour la communauté d'utilisateurs ayant besoin d'accéder aux ressources de calcul distribuées. DIRAC forme une couche entre une communauté particulière et diverses ressources informatiques pour permettre une utilisation optimisée, transparente et fiable[43]. Il a été le premier système de soumission de jobs pilotes dès le projet DataGrid au début des années 2000.

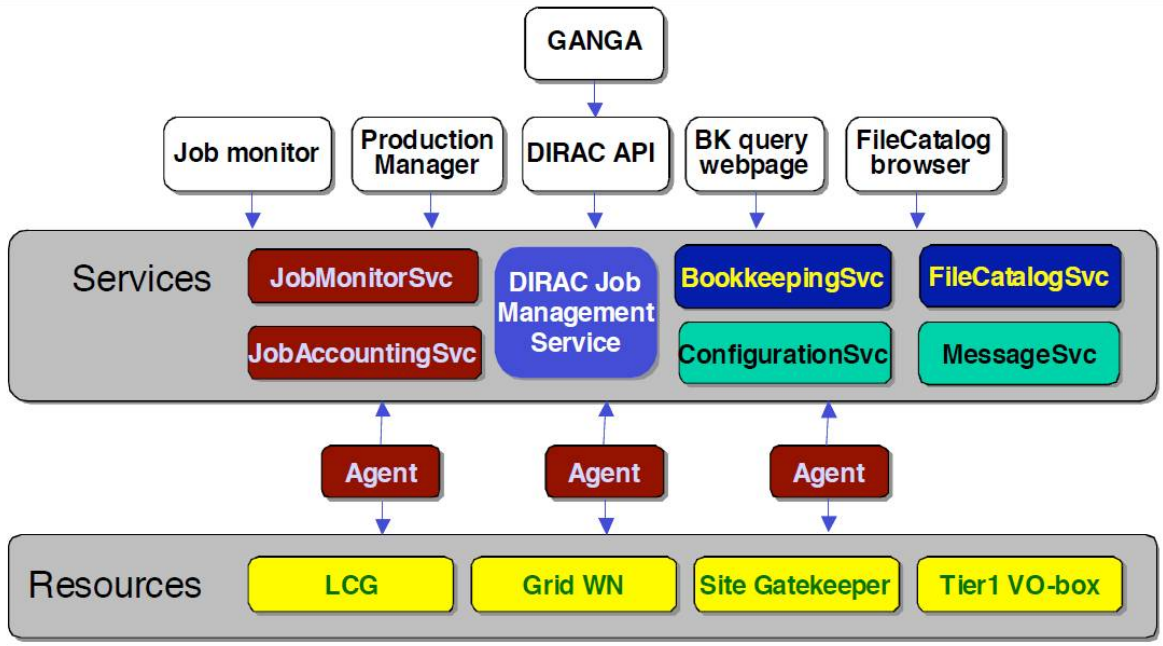


Figure 2-13: L'architecture de la plate-forme DIRAC (Source : [43])

DIRAC suit le paradigme de l'architecture orientée services (SOA). Ses principales composantes sont présentées dans la Figure 2-13. Les composants de DIRAC peuvent être regroupés en quatre groupes (ressources, services, agents et Interfaces):

- les ressources: Les ressources sont des composants qui permettent d'accéder au calcul et au stockage disponibles de DIRAC
- Service: Le système de DIRAC est construit autour d'un ensemble de services faiblement couplés qui maintiennent l'état du système et aident à la gestion de la charge et des données. Les services sont des composants passifs qui ne réagissent aux demandes de leurs clients pour solliciter d'autres services afin d'accomplir les demandes.
- Agent : les agents sont des composants logiciels légers et faciles à déployer qui exécutent des processus indépendants.
- Interface: les fonctionnalités de la plate-forme DIRAC sont exposées aux développeurs du système et aux utilisateurs de différentes façons. Pour les développeurs faisant usage du système de DIRAC, la fonctionnalité est fournie comme une API Python. Pour les utilisateurs du système de DIRAC la fonctionnalité est accessible par une interface en ligne de commande.

DIRAC fournit également des interfaces Web pour les utilisateurs et les gestionnaires du système pour surveiller le comportement du système et pour contrôler les tâches en cours.

DIRAC déploie des agents pilotes sur des nœuds de calcul (Worker Nodes) comme des jobs réguliers en utilisant le mécanisme de soumission des jobs standard de la grille. Il forme un système de gestion de la charge distribuée et réserve les ressources pour une utilisation immédiate.

Le système de production distribué DIRAC fournit les services suivants:

- Définition des tâches de production
- Installation de logiciels sur des sites de production
- Planification et surveillance des jobsGestion et réplication de données

Pour un système de production, le composant en charge de la planification des jobs (job scheduling) est très important. Deux paradigmes différents peuvent être choisis :

Le paradigme PUSH : le planificateur (scheduler) utilise les informations sur la disponibilité et l'état des ressources de calcul afin de trouver le meilleur CE selon les exigences particulières d'un job.

Dans le paradigme PULL, en revanche, c'est la ressource de calcul qui cherche des tâches à exécuter. Les jobs sont d'abord accumulés par un service de production, validés et mis dans une file d'attente. Lorsqu'une ressource de calcul est disponible, elle envoie une requête pour un job à exécuter au service de production. Le service de production choisit un job, selon les capacités de la ressource et le lui envoie pour qu'il puisse s'y exécuter.

Il y a des avantages et des inconvénients dans les deux approches. Dans le paradigme PUSH, l'information sur l'état dynamique de toutes les ressources est généralement collectée dans un seul endroit. Pour chaque job, le meilleur choix possible peut être fait.

Dans le paradigme PULL, la ressource la plus performante est la plus sollicitée. Par conséquent, l'équilibrage de la charge est atteint plus aisément.

DIRAC, comme d'autres systèmes de production de la grille de calcul (DIANE ou WPE) est développé avec le paradigme PULL. L'architecture de DIRAC se compose de services centraux: Production, Monitoring et Bookkeeping. Les agents de production s´exécutent en permanence sur chaque site de production. Les services centraux ont pour rôle de préparer les productions, surveiller l´exécution des jobs et gérer les paramètres des jobs. Les agents examinent l'état des queues de production locales. Si les ressources locales sont capables

d'accepter la charge, l'agent envoie une demande de job au service de production centrale et assure l'exécution et la surveillance de ce job. Après l´exécution du job, l'agent transfère les données de sortie et met à jour le catalogue de métadonnées.

Tous les utilisateurs sont affectés aux groupes de DIRAC et les groupes peuvent être affectés à une Organisation Virtuelle. Les utilisateurs d'un groupe partagent le quota de ressources de leur groupe. Lorsqu'il y a plusieurs lots de tâches à exécuter au sein d'un groupe, la politique de l'ordonnancement entre les utilisateurs d'un groupe est de type Round Robin.

DIRAC est maintenant largement utilisé sur de nombreuses infrastructures dans le monde entier. Son concept permet d'agréger dans un seul système informatique des ressources de nature différente, tels que des grilles de calcul, des clouds ou des clusters, et ce de façon transparente aux utilisateurs.

2.5 Conclusion

Dans ce chapitre, nous avons présenté les enjeux de la recherche de nouveaux médicaments issus de la biodiversité au Vietnam. Nous avons montré comment les infrastructures distribuées de grille constituaient des environnements pertinents pour rechercher *in silico* les cibles biologiques sur lesquelles les composés isolés par l'INPC pouvaient présenter une activité. Nous avons ensuite présenté comment l'utilisation de plates-formes avait permis l'utilisation massive des grilles dans le déploiement de calculs d'ancrage à grande échelle.

L'arrivée de la technologie des clouds apporte de nouvelles opportunités, notamment en termes de flexibilité des systèmes d'exploitation ou de gestion des pics de charge. Mais l'offre des clouds académiques est encore limitée et un long chemin reste à parcourir avant que celle-ci ne soit comparable à celle des grilles. Des efforts importants sont notamment nécessaires pour fédérer les ressources des clouds académiques au niveau national et international.

Dans ce contexte, l'utilisation de plates-formes pour gérer les projets de recherche de nouveaux médicaments semblables à celui de l'INPC demeure une voie privilégiée. Ces plates-formes constituent des couches intermédiaires entre les utilisateurs et les ressources de la grille et des clouds qui permettent de masquer les évolutions technologiques. De plus en plus d'équipes de recherche accèdent aux ressources de la grille à travers de telles plates-

formes et elles sont identifiées comme des outils privilégiés pour exposer les ressources d'une fédération de clouds de façon homogène aux utilisateurs.

L'opération d'une telle plate-forme à l'INPC ne semble pas une approche pertinente car son installation, sa configuration et sa maintenance requièrent des compétences pointues en informatique. Les plates-formes utilisées actuellement en France, en Corée et à Taïwan sont opérées dans des centres de calcul nationaux. Cependant, il peut être envisagé que les chercheurs de l'INPC soient clients de telles plates-formes, mais se pose alors la question de la gestion des clients de ces plates-formes.

Dans le prochain chapitre, nous allons donc nous intéresser à l'ordonnancement des tâches soumises à travers des plates-formes sur grille et cloud.

CHAPITRE 3 : ETAT DE L'ART DE L'ORDONNANCEMENT DES PLATES-FORMES SUR GRILLE ET CLOUD

3.1 Introduction

Nous avons décrit dans le chapitre précédent le besoin des chercheurs de l'INPC de lancer de façon massive et répétitive des tâches de docking pour sélectionner les cibles biologiques potentiellement inhibées par les ligands isolés à partir de la biodiversité en Vietnam.

Nous avons présenté les scénarii actuels d'utilisation des ressources distribuées offertes par les infrastructures de grille et de cloud pour les sciences du vivant et la recherche de nouveaux médicaments. Nous avons conclu que l'utilisation d'une plate-forme permettait de faciliter le déploiement des calculs à grande échelle et de masquer les difficultés techniques liées à la migration de la grille vers le cloud. Nous avons aussi identifié la difficulté pour l'INPC de gérer sa propre plate-forme dédiée et la nécessité de partager cette plate-forme avec d'autres utilisateurs. Deux options existent :

- Soit la plate-forme partagée est dédiée à la recherche de nouveaux médicaments et le partage s'effectue entre biochimistes utilisant les mêmes logiciels.
- Soit la plate-forme est pluridisciplinaire, partagée entre plusieurs communautés d'utilisateurs.

La première approche est celle utilisée par exemple dans le cadre du projet WeNMR (http://www.weNMR.eu) qui propose à la communauté des chercheurs utilisant la Résonance Magnétique Nucléaire en biologie structurale la possibilité de faire tourner un ensemble de logiciels de référence d'analyse des données NMR de structure sur la grille[44]. Ces logiciels sont disponibles à travers un portail métier dont le succès est maintenant considérable : près de 600 utilisateurs enregistrés ont ainsi soumis près de 2 millions de jobs et consommé près

de 5 millions d'heures CPU sur les 90.000 processeurs de l'organisation virtuelle WeNMR d'Octobre 2012 à Septembre 2013.

Un autre exemple est le portail VIP (http://www.creatis.insa-lyon.fr/vip) dédié à l'imagerie médicale [45]. La figure 3.1 illustre l'évolution de l'utilisation des ressources de la grille (exprimée en années CPU cumulées) de Novembre 2012 à Octobre 2013 à travers le portail VIP.

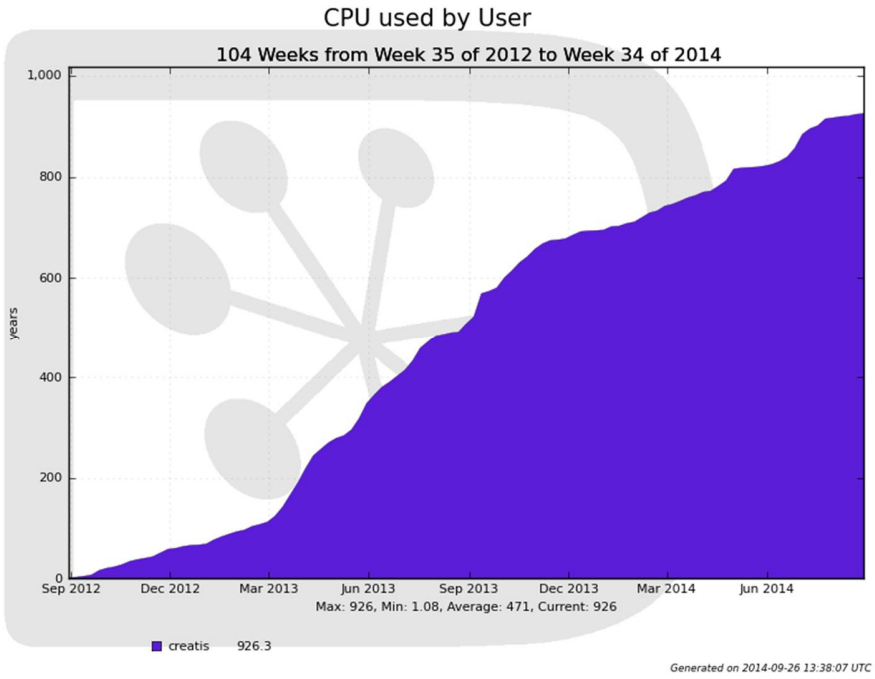


Figure 3-1: Evolution de la consommation des ressources de la grille à travers la plate-forme VIP de Novembre 2012 à Septembre 2014 [45]

Plus de 430 années CPU ont été consommées de Septembre 2013 à Août 2014, soit de l'ordre de 25% de la consommation des ressources de l'organisation virtuelle Biomed. Le portail VIP utilise la plate-forme DIRAC pour la distribution des tâches de ses utilisateurs sur la grille.

La deuxième approche est celle adoptée par France Grilles qui a mis à disposition des communautés une plate-forme DIRAC ouverte. Les utilisateurs de cette plate-forme soumettent des tâches sur différentes organisations virtuelles (référence FG-DIRAC). Le serveur est hébergé au Centre de Calculs de l'IN2P3. Là aussi, le succès a été tout de suite au rendez-vous. Aujourd'hui ce serveur a 180 utilisateurs enregistrés. La figure 3.2 montre la consommation des ressources par groupe d'utilisateurs tirée du site FG-DIRAC (https://dirac.france-grilles.fr)

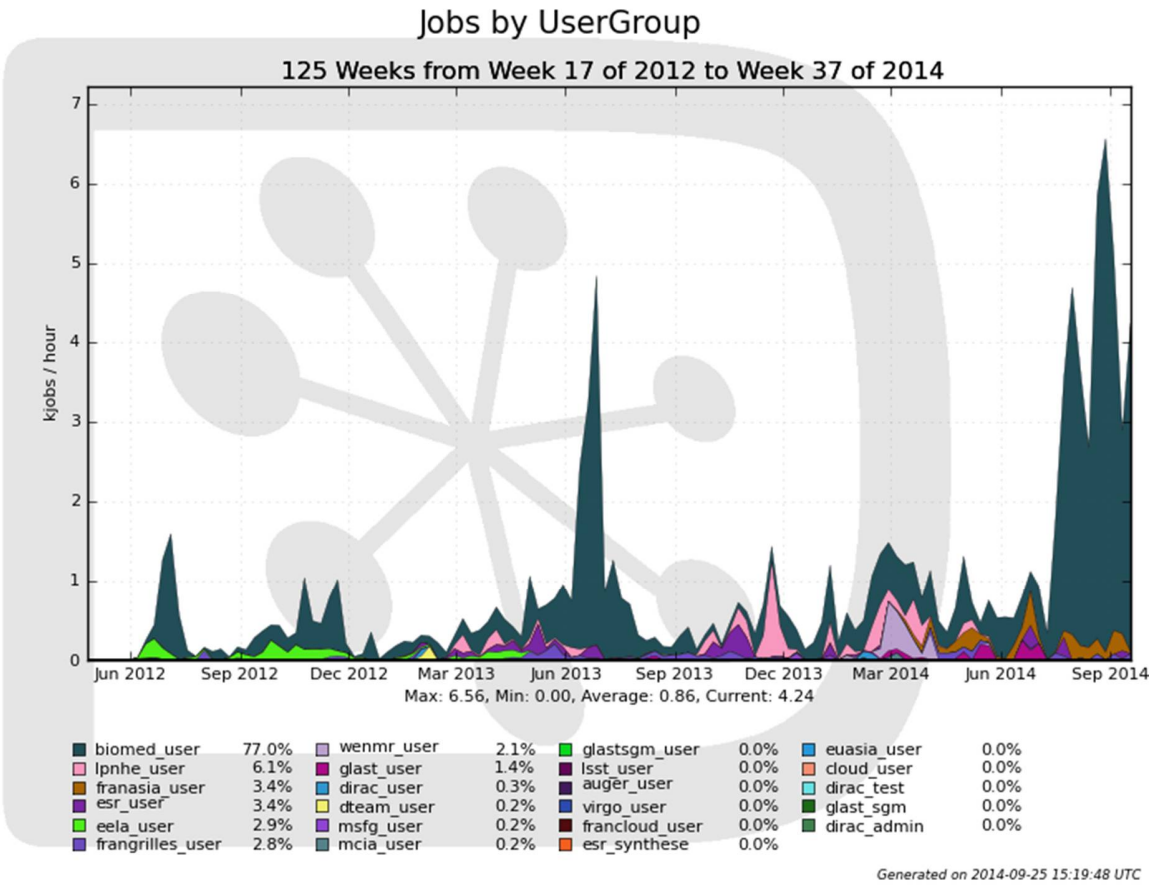


Figure 3-2: Statistiques d'utilisation du serveur FG-DIRAC par les différents groupes de Juin 2012 à Septembre 2014

Le succès de ces outils illustre l'évolution des comportements des utilisateurs de la grille. De plus en plus utilisent des portes d'entrée scientifiques à la grille (en anglais « scientific gateways ») comme VIP ou WeNMR adossés à des plates-formes comme DIRAC ou FG-DIRAC sans se préoccuper de l'endroit où leurs calculs sont exécutés. La plupart ne sont plus titulaires de certificat, se contentant d'utiliser le certificat robot du portail.

Par exemple, seulement une petite fraction des utilisateurs de VIP (27 sur 479 en Novembre 2013) sont enregistrés sur la VO Biomed tandis que les autres utilisent un certificat robot.

Le succès de ces portails et de ces plates-formes et la croissance rapide de leur adoption ouvre la question de la gestion des utilisateurs. Quelle politique adopter pour gérer les demandes des utilisateurs ? Nous allons dans ce chapitre présenter les enjeux de l'allocation des ressources et de l'ordonnancement des tâches pour une plate-forme sur la grille et le cloud. Nous introduirons la notion d'équité entre les utilisateurs et présenterons des quantités permettant de mesurer la satisfaction des utilisateurs. Celles-ci seront utilisées comme critères d'optimisation des politiques d'ordonnancement étudiées dans la suite de cette thèse.

3.2 Ordonnancement équitable des ressources

3.2.1 Notion d'équité

Le libre accès à des ressources limitées entraîne inévitablement une utilisation inefficace qui pénalise l'ensemble des participants. En effet, chaque individu cherche à maximiser son intérêt personnel et aura tendance à s'approprier de manière excessive les ressources au détriment des autres [46]. Une solution pour faire face à ce problème consiste à organiser la gestion de ces ressources. Cette gestion doit proposer des mécanismes de régulation afin de garantir une part équitable aux différents utilisateurs.

L'équité est un important facteur d'évaluation dans tous les problèmes d'allocation, de distribution ou de partage de ressources entre utilisateurs ; ceci inclut entre autres les systèmes distribués comme les grilles de calcul.

Ainsi, un ordonnancement équitable envers les utilisateurs encourage leur participation, alors qu'un ordonnancement inéquitable provoque l'abandon de la participation. De plus, des méthodes capables de justifier les choix effectués et de montrer que le traitement est équitable, permettent de régler les conflits.

Il n'y a de problèmes d'équité que lorsque les demandes excèdent les ressources disponibles. Dans ce cas de figure, la mise en place d'une politique d'allocation des ressources est nécessaire pour départager les utilisateurs. Ces politiques d'allocations de ressources doivent obéir à certaines règles, ce qui permet de les automatiser, de les évaluer et de les vérifier.

Lorsque les ressources ne sont pas suffisantes pour l'ensemble des utilisateurs et qu'un conflit survient, la politique d'allocation fait alors office de médiateur [47] et répartit les ressources entre les utilisateurs.

La grille de calcul permet le partage de ressources de calcul et/ou de stockage. Dans un tel environnement apparaît le problème de distribuer les ressources à la fois de façon efficace et équitable. Un des principes importants est que la grille doit respecter la liberté des fournisseurs. Les sites ont la liberté d'attribuer leurs ressources comme ils le souhaitent à des

groupes particuliers, en revanche une fois l'attribution décidée, on doit respecter le principe d'équité entre individus égaux en droits.

En conséquence, il existe dans une grille plusieurs niveaux d'ordonnancement :

- au niveau central, les utilisateurs soumettent leurs tâches à un système de gestion de charge (en anglais "Workload Management System (WMS)") dont le rôle est de répartir les tâches sur les sites [10]. Un site est passif devant l'envoi des tâches et ne participe pas directement à la sélection par un WMS : les sites publient de l'information sur l'état de leurs clusters et le WMS gère la sélection des sites pour chacune des tâches. L'utilisateur peut spécifier certaines contraintes à respecter pour la sélection du site dans son JDL [48]
- au niveau de chaque site de la grille, le système de batch est chargé de la gestion de l'envoi des tâches depuis une machine frontale, le "Computing Element (CE)", sur l'ensemble des machines de calcul, les "Worker Nodes (WN)". Le système de batch ne décide pas du placement ; c'est le rôle de l'ordonnanceur. Couramment utilisé sur les clusters intégrés dans la grille, l'ordonnanceur MAUI [49] fonctionne sur le principe d'une file d'attente triée selon la priorité des tâches. L'administrateur peut spécifier des pondérations selon les groupes, les utilisateurs, le temps d'attente. Il est aussi possible d'attribuer une priorité supplémentaire à une tâche, par exemple lorsque son groupe ou son utilisateur a consommé un temps de calcul inférieur à son quota de temps pendant une période. Nous reviendrons dans ce chapitre sur cette fonctionnalité, appelée "Fair Share".
- Eventuellement, un troisième niveau d'ordonnancement est celui de la plate-forme. Il s'agit de définir une politique pour créer les jobs pilotes et définir un ordre de traitement par ces jobs pilotes des tâches soumises par les utilisateurs de la plate-forme. L'ordonnancement va permettre d'organiser la file d'attente des utilisateurs si ceux-ci partagent les agents (DIRAC) ou gérer la création des agents par utilisateur quand les agents sont privés (HTCaaS)

Nous allons maintenant voir quelques politiques d'ordonnancement couramment utilisées pour la gestion des ressources et des files d'attente.

3.2.2 Politiques d'ordonnancement

Nous n'allons présenter dans ce paragraphe que les algorithmes d'ordonnancement les plus connus et les plus couramment utilisés, notamment sur la grille. Traditionnellement, les ordonnanceurs sont conçus pour affecter les ressources de façon uniforme aux processus (dans l'idéal, tous les processus s'exécuteraient en même temps et à la même vitesse). Cependant, ce type d'ordonnancement a tendance à favoriser les utilisateurs qui lancent beaucoup de processus en allouant le CPU proportionnellement au nombre de processus et risque d'étouffer les utilisateurs avec peu de tâches. A l'inverse, un ordonnanceur qui privilégie l'équilibre entre les utilisateurs peut pénaliser les très gros consommateurs de ressources, ce qui pose problème quand ceux-ci travaillent pour la collectivité. C'est ainsi le cas pour les expériences au LHC où la production de données simulées est centralisée afin d'être plus efficace. Satisfaire tous les utilisateurs est donc la quadrature du cercle et les politiques d'ordonnancement décrites ci-dessous vont finalement privilégier une catégorie d'utilisateurs.

3.2.2.1 First In First Out (FIFO)

L'ordonnancement est fait dans l'ordre d'arrivée en gérant une file unique des processus sans priorité ni préemption: chaque processus s'exécute jusqu'à son terme ; le processus élu est celui qui est en tête de liste, le premier arrivé. Cet algorithme est facile à implanter, mais il est loin d'optimiser le temps de traitement moyen et il n'est pas très équitable.

La plate-forme WISDOM [28], développée par le Laboratoire de Physique Corpusculaire (LPC) Clermont-Ferrand, France, pour les projets WISDOM-I (2005) et WISDOM-II (2006) utilisait une politique d'ordonnancement FIFO.

3.2.2.2 Round robin

Il s'agit d'un algorithme ancien, simple et fiable. Les jobs sont servis tour à tour : dans le cas avec préemption des jobs, chaque job est exécuté dans une tranche de temps constante (quantum). A la fin du quantum, si le job n'est pas encore terminé, il est remis dans la file d'attente et laisse ainsi la machine au job suivant. Sans préemption des jobs, l'ordonnanceur sélectionne tour à tour un job d'un utilisateur différent dans la file d'attente.

La plate-forme DIRAC [43] utilise la politique d'ordonnancement Round Robin sans préemption.

3.2.2.3 Ordonnancement équitable ou « fairshare scheduling »

Cette méthode d'ordonnancement a pour objectif que chaque utilisateur obtienne une part équitable plutôt que chaque processus [50]. Chaque utilisateur a un pourcentage du temps de calcul qu'il divise entre ses tâches. Il est aussi possible d'attribuer une priorité supplémentaire à une tâche, par exemple lorsque son groupe ou son utilisateur a consommé un temps de calcul inférieur à son quota de temps pendant une période. Une variable baptisée FS est calculée pour chaque utilisateur. Quand un processeur est libre, c'est l'utilisateur avec le plus petit FS qui peut utiliser le processeur. FS est calculé avec la formule suivante (2.1) :

$$FS_j = \sum_{i=0}^{Max\_depth} N_i^j * (Decay\_rate)^i \qquad (2.1)$$

Avec :

- le paramètre « **Max_depth** », nombre d'intervalles enregistrés dans la table de l'histoire.
- N_i^j le paramètre « **Decay_rate** » est le nombre de tâches d'utilisateur j qui sont exécutées dans l'intervalle i
- la durée de chaque intervalle est fixée par le le paramètre « **Duration** »

Considérons par exemple deux utilisateurs avec un historique d'utilisation de ressources. Pour un choix de paramètres (Duration = 1h, Max_depth = 5, Decay_rate=0.85), on obtient le tableau 3-1 et des valeurs FS qui attribuent la priorité à l'utilisateur 1 pour le prochain processeur disponible.

	0	**1**	**2**	**3**	**4**	**5**	**FS**
User 1	2	3	2	1	5	2	10.107
User 2	1	4	4	5	6	8	17.042

Table 3-1: Exemple d'historique d'utilisation des ressources et de calcul de FS

L'algorithme Fair Share est utilisé pour la planification dans MAUI [49] (http://supercluster.org/projects/maui), gestionnaire de tâches hautement configurable et efficace qui a reçu une large acceptation dans la communauté HPC. Il est actuellement utilisé sur des centaines de systèmes IBM SP-2, SGI Origin 2000, et des clusters Linux à travers le

monde [49]. Cet algorithme suit l'utilisation des ressources dans le temps et l'utilisation de ces données historiques pour régler le comportement d'ordonnancement.

De même, l'algorithme d'ordonnancement équitable de la plate-forme HTCaaS utilise une table de l'histoire de l'utilisation des ressources par chaque utilisateur.

3.2.2.4 Longest Processing Time (LPT)

Les jobs sont triés par ordre de durée décroissante : les jobs les plus longs seront exécutés en premier. Cet algorithme a une performance garantie pour le problème du Makespan[51].

3.2.2.5 Shortest Processing Time (SPT)

Les jobs sont triés par ordre de durée croissante : les jobs les plus courts seront exécutés en premier.

3.2.3 Critères d'évaluation des politiques d'ordonnancement

Afin d'évaluer les politiques d'ordonnancement, on doit introduire les critères d'optimisation. Le concept de fonction objectif est utilisé en optimisation mathématique pour désigner une fonction qui sert de critère pour déterminer la meilleure solution à un problème d'optimisation. Concrètement, elle associe une valeur à une instance d'un problème d'optimisation. Le but du problème d'optimisation est alors de minimiser ou de maximiser cette fonction jusqu'à l'optimum, par différents procédés.

Pour introduire les différents critères d'optimisation, nous allons introduire la notation suivante : Soit une série de N jobs, J_1, J_2, … J_N soumis par un utilisateur. Chaque job J_i arrive à l'instant r_i et son exécution est finie à l'instant C_i.

Makespan

La fonction objectif la plus étudiée dans la littérature de l'ordonnancement parallèle est le makespan ou temps de complétion maximal : $C_{max} = \max (C_i)$

La minimisation du makespan est conceptuellement une approche basée sur système, visant à assurer l'utilisation efficace de la plate-forme.

Temps de réponse

Les utilisateurs individuels sont généralement plus intéressés par des mesures basées sur les jobs, tels que le temps de réponse ou temps d'attente (flowtime en anglais) qui représente le temps d'un travail individuel passé dans le système. Avec les notations définies précédemment, la formule du temps de réponse F_i du job J_i s'écrit : $F_i = C_i - r_i$

L'optimisation du temps total de réponse, $\Sigma_j F_j$, souffre de la limitation que l'exécution de certains jobs peut être retardée de façon illimitée, par manque de ressources [52]. En revanche, la minimisation du temps de débit maximal, $\max_j F_j$, ne souffre pas de cette limitation, mais elle tend à favoriser les jobs de longue durée au détriment de ceux de courte durée. Pour surmonter ce problème, une approche commune [53] met l'accent sur le temps de réponse pondéré (weighted flowtime) qui prend en compte la durée des jobs pour compenser le biais introduit par les jobs de courte durée.

La somme des temps de réponse pondérés et le maximum des temps de réponse pondérés peuvent alors être définis de façon analogue. Notons cependant que le problème de famine de ressources, identifié pour la minimisation de la somme des flowtime est inhérent à toutes les fonctions objectif utilisant la somme, de sorte que la somme « weighted flowtime » souffre de la même faiblesse.

Ralentissement

Le ralentissement ou « stretch » en anglais est un cas particulier de temps de réponse pondéré, dans lequel le poids d'un job est inversement proportionnel à sa durée d'exécution [52]. C'est donc une mesure assez juste du niveau de service fourni à une tâche particulière et est plus pertinent que le temps de réponse dans un système avec des tailles de job très variables. Par conséquent, nous nous concentrerons principalement sur l'utilisation du stretch comme métrique pour caractériser l'expérience de l'utilisateur sur la plate-forme.

Table 3-2: Exemple du stretch de 2 utilisateurs

La Figure 3-2 illustre le sens du stretch dans le cas de deux utilisateurs qui envoient des jobs à une plate-forme. L'utilisateur 1 envoie un job de durée 10 secondes qui atteint la plate-forme à

t = 0s. L'utilisateur 2 a un job de durée 5 secondes qui atteint la plate-forme à t = 5s. La tâche de l'utilisateur 1 est exécutée avant celle de l'utilisateur 2. Comme l'illustre la Figure 3-2, la tâche de l'utilisateur 1 se termine à t = 10s. Parce que l'utilisateur 2 doit attendre la complétion de la tâche de l'utilisateur 1, l'exécution de sa tâche se termine à t = 15s. Bien que les deux utilisateurs aient passé le même temps sur la plate-forme (10s), la satisfaction de l'utilisateur 2 est moindre que celle de l'utilisateur 1. Le ralentissement de l'utilisateur 1 est égal à 10/10 = 1, tandis que le stretch pour l'utilisateur 2 est égal à 10/5 = 2. Cet exemple illustre pourquoi le stretch S mesure l'expérience de l'utilisateur sur la plate-forme. Plus le stretch est petit, meilleure est la satisfaction de l'utilisateur.

Nous nous intéressons maintenant à un groupe de K utilisateurs. Chaque utilisateur U_i (i=1…K) soumet un ensemble de N_i jobs. Le temps d'exécution d'un job j de l'utilisateur i sur un ordinateur est dénoté par P_j^i (j=1 … N_i). L'ensemble des jobs arrive à la plate-forme au temps r_i. Le Job j de l'utilisateur i est fini au temps C_j^i . Le temps pour finir tout le projet est $C_i^{max} = max_{j=1..N_i} C_j^i$. Le ralentisement du projet de l'utilisateur Ui est défini par la formule:

$$S_i = \frac{C_i^{max} - r_i}{\sum_{j=1}^{N_i} P_j}$$

Deux critères peuvent être utilisés pour évaluer l'efficacité de la politique d'ordonnancement des tâches de la plate-forme :

- Stretch max (S_{max} = max S_i) donne la plus mauvaise expérience d'un utilisateur du système.
- Strech moyen (S_{moyen} = <S_i>) donne une indication de l'efficacité de l'algorithme sur le ralentissement expérimenté en moyenne par les utilisateurs.

De nombreuses équipes ont mené des recherches sur l'optimisation centrée sur le système dans le contexte de machines dédiées (c'est à dire machines toujours disponibles). S. Muthukrishnan et al. [54] ont exploré les performances de la politique Shortest Processing Time sur mono et multiprocesseurs pour optimiser la moyenne du ralentissement. Legrand et al. ont montré dans [55] que la politique SPT est très efficace pour optimiser les stretchs max et moyen.

Dans cette thèse, nous nous intéressons à minimiser à la fois les stretchs max et moyen des projets d'utilisateurs. Notre problème est donc une optimisation sur ces deux critères.

Nous allons maintenant nous intéresser aux spécificités des problématiques d'ordonnancement d'une plate-forme déployant des jobs pilotes sur grille et sur cloud.

3.3 Ordonnancement de plates-formes sur grille et sur cloud

Le travail décrit dans ce manuscrit s'inscrit dans la continuité du travail accompli dans le cadre de la thèse d'Emmanuel Medernach [56] dédiée à l'allocation de ressources et l'ordonnancement multi-utilisateurs au niveau local d'un nœud de la grille, pour satisfaire à la fois des critères d'équité et d'efficacité.

La problématique à laquelle nous nous intéressons dans cette thèse est l'ordonnancement des tâches reçues par une plate-forme gérant des jobs pilotes sur la grille et sur le cloud.

Comme nous l'avons déjà évoqué, il existe plusieurs niveaux d'ordonnancement sur la grille. L'ordonnancement par les services centraux de la grille des tâches des utilisateurs est implémenté selon le modèle PUSH : l'utilisateur envoie ses tâches au Workload Management System (WMS). Le WMS trouve un nœud de calcul (Worker Node) approprié. Ensuite le WMS va « pousser (push) » la tâche sur le Worker Node qui l'exécutera (figure 3.3).

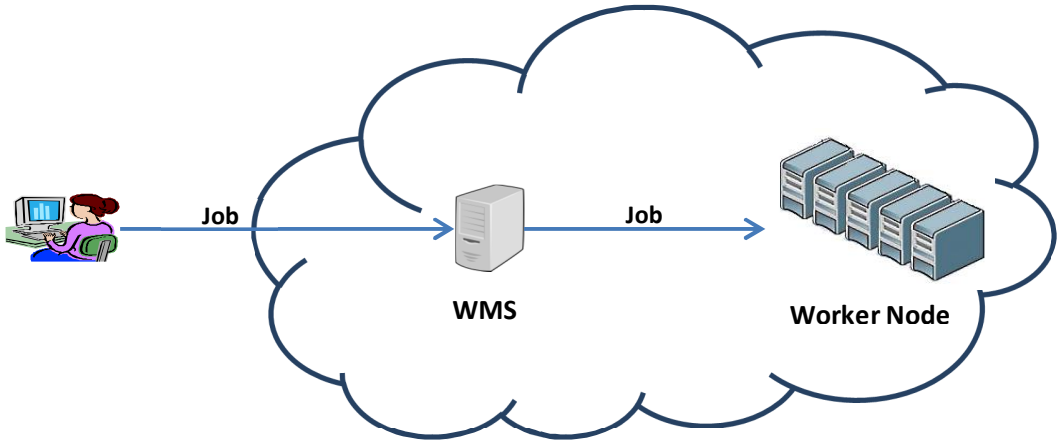


Figure 3-3: Le modèle PUSH de la grille

En revanche, les plates-formes basées sur les agents pilotes (pilot agent en anglais) utilisent le modèle PULL pour la soumission efficace et le contrôle d'un grand nombre de jobs sur la grille. Au lieu de laisser à l'utilisateur le soin de soumettre directement leurs jobs sur la grille

via le modèle PUSH du WMS, la plate-forme soumet automatiquement des jobs dits génériques sur la grille comme des jobs normaux. Les agents pilotes sont génériques car leur fonction n'est que d'exécuter les tâches des utilisateurs (souvent de petits programmes mais exécutés en grande quantité). Ces tâches sont mises dans des files d'attente et ensuite récupérées (pull) par des agents pilotes en cours d'exécution sur un Worker Node pour être exécutées. L'agent pilote lui-même est un job régulier de grille, créé par un gestionnaire de ressources de grille, soumis automatiquement par la plate-forme et exécuté sur un processeur quelque part sur la grille. Le modèle PULL est bien adapté aux propriétés des infrastructures de la grille: une machine plus rapide récupérera plus de tâches que les autres, les tâches ayant échoué peuvent être resoumises facilement et la latence est ainsi réduite.

3.3.1 Ordonnancement au niveau de la plate-forme

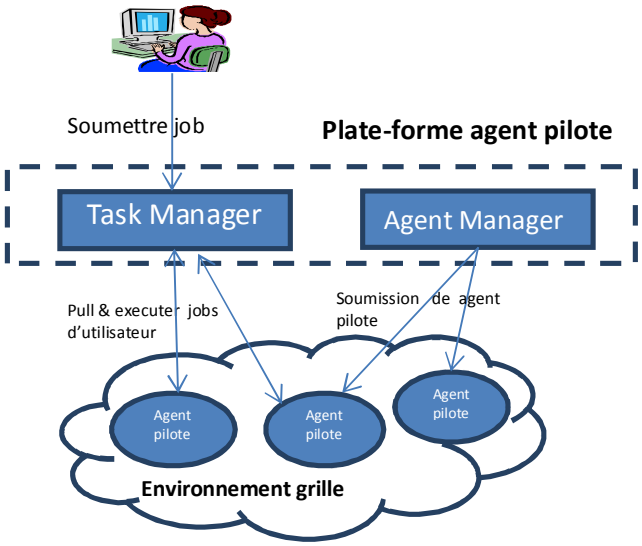


Figure 3-4: Le modèle PULL de la plate-forme agent-pilote

Comme montrée dans la figure 3.4, une plate-forme à agents pilotes dispose de deux modules principaux, le gestionnaire de tâches (Task Manager en anglais) et le gestionnaire d'agents (Agent Manager en anglais) :

- Le Gestionnaire d'Agent soumet automatiquement des agents pilotes à l'infrastructure de la grille/cloud. Si le Gestionnaire de Tâches a des tâches d'utilisateur dans sa file d'attente, l'agent pilote télécharge une tâche et l'exécute.

- Le Gestionnaire de Tâche reçoit les tâches de l'utilisateur et contrôle les files d'attente de tâches. Lors de la réception d'une demande d'un agent pilote, il choisit une tâche dans les files d'attente et l'envoie à l'agent pilote demandé

Nous voyons donc que l'ordonnancement des tâches des utilisateurs a lieu à deux niveaux comme le montre la figure 3-5:

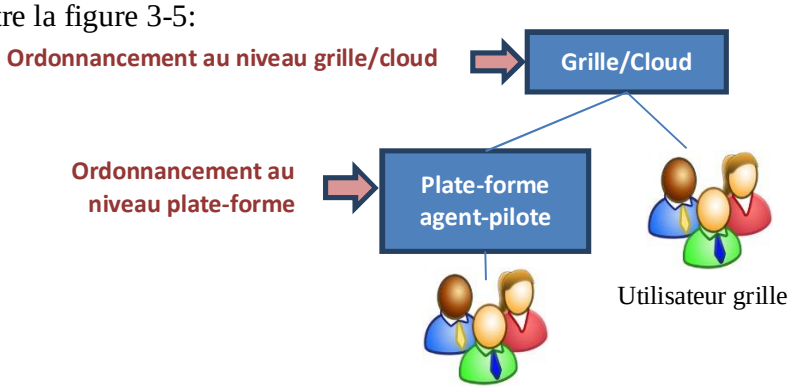


Figure 3-5: Deux niveaux d'ordonnancement dans la plate-forme agent pilote

L'ordonnancement a lieu au niveau de la grille/cloud lorsque l'agent pilote est envoyé à partir de la plate-forme à l'ordonnanceur de grille dans une file d'attente. L'ordonnancement au niveau de la grille est impacté par les configurations choisies par les administrateurs du site. La théorie de l'ordonnancement sur des systèmes distribués a fait l'objet de travaux approfondis [56], [57].

L'ordonnancement au niveau de la plate-forme est en général effectué par le Gestionnaire de Tâches. Par exemple lorsqu'un utilisateur envoie un projet de criblage virtuel à celui-ci, ses tâches de docking sont mises en file d'attente des tâches. L'ordonnanceur de la plate-forme calcule les priorités des tâches, et répond à la demande des agents pilotes en leur envoyant les tâches qui sont classés avec la plus haute priorité en fonction de sa politique d'ordonnancement. Nous nous occupons de l'ordonnancement au niveau de la plate-forme et proposons différentes politiques de l'ordonnancement.

Dans le cas particulier où les agents sont privés, c'est-à-dire réservés à un utilisateur comme dans le cas de la plate-forme HTCaaS, c'est au niveau du gestionnaire d'agents que l'ordonnancement va gérer la création des agents pour chaque utilisateur.

Un mécanisme d'appariement permet d'associer une tâche d'un utilisateur à un agent pilote dans le gestionnaire de tâche. Il y a deux files d'attente dans le gestionnaire de tâches : une file d'attente de jobs d'utilisateurs et une file d'attente de nœuds de calculs (Figure 3-6). L'algorithme 3.1 définit l'algorithme de Matcher dans le module de gestion de tâches.

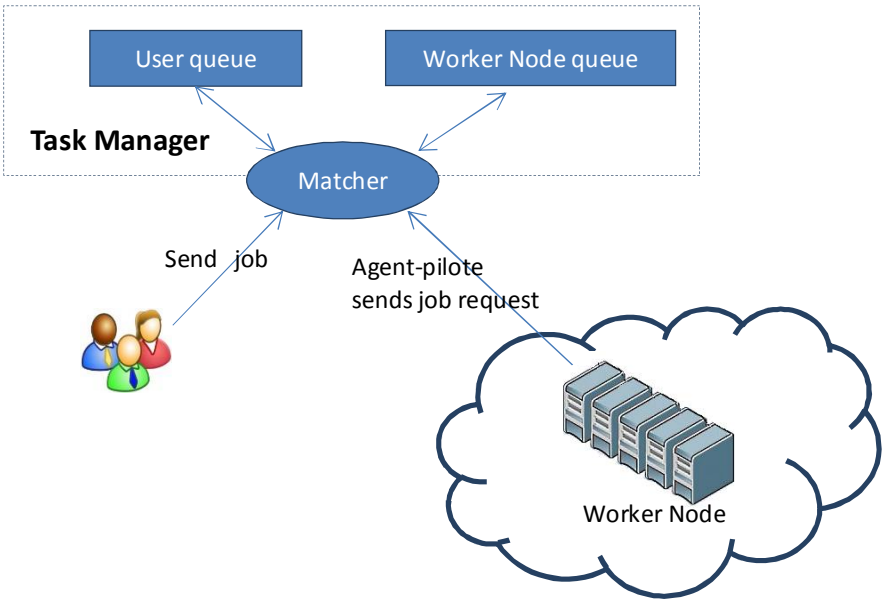


Figure 3-6: Deux files d'attente dans le Task Manager

Algorithme 3.1: Algorithme de Matcher

WHILE (Task Manager is alive)
DO
    Receive request

    IF (Request is from Worker Node) **THEN**
        IF (there is job in Job Queue) **THEN**
            Get the first job in queue and send to Worker Node
        ELSE
            Put Worker Node in Worker Node queue
        ENDIF
    ENDIF

    IF (Request is from User) **THEN**
        IF (there is Worker Node in Worker Node queue) **THEN**
            Get the first Worker Node in queue and send job to it
        ELSE
            Put job in the Job queue
        ENDIF
    ENDIF
END WHILE

Dans la plate-forme à agents pilotes, l'utilisateur soumet ses tâches au gestionnaire de tâches. Ces jobs vont être mis dans la file d'attente. Quand un agent pilote envoie la requête de tâche au gestionnaire de tâches, ce dernier va trouver une tâche dans la file d'attente des tâches et l'envoyer à l'agent pilote.

Du point de vue de la plate-forme, les agents pilotes sont comme des machines : les jobs d'utilisateur sont assignés aux agents pilotes par le « Matcher ». Les jobs d'utilisateur sont reçus et mis dans la file d'attente contrôlée par l'ordonnanceur du gestionnaire de tâches.

L'ordonnancement des tâches au niveau de la plate-forme s'inscrit dans un contexte de disponibilité limitée des ressources ; en effet, la durée de vie des agents est limitée. En général, ceux-ci s'exécutent pendant la durée maximum autorisée sur les sites de la grille.

Dans [58], les auteurs ont examiné quelques algorithmes d'ordonnancement dans ce contexte particulier de disponibilité limitée de la machine. La politique Longest Processing Time est l'un des algorithmes d'ordonnancement en ligne proposés dans cette recherche. Cependant, ces recherches sont effectuées sur les paramètres du système central (makespan, somme des temps de la fin, le retard max, ...).

Nous allons présenter un état de l'art de l'ordonnancement au niveau des plates-formes dans la section suivante.

3.3.2 Etat de l'art de l'ordonnancement au niveau des plates-formes

3.3.2.1 DIET

La plate-forme DIET [59] utilise les deux modèles « PUSH » et «PULL». Les demandes de ressource sont poussées par les clients de plate-forme, alors que les ressources tirent les options de demandes de clients. Chaque client et serveur contribuent à la prise de décision d'ordonnancement. Cependant, les autres plates-formes à agents pilotes prennent la plupart des décisions d'ordonnancement de manière centralisée. Par conséquent, la politique d'ordonnancement de DIET n'est pas directement applicable à notre énoncé du problème.

3.3.2.2 AppLeS

Dans [60], les auteurs présentent une solution de l'ordonnancement au niveau de l'application appelée AppLeS. Ils décrivent une approche spécifique de l'ordonnancement des applications parallèles individuelles sur les systèmes de production hétérogènes, en utilisant des informations complètes sur l'application et des ressources pour optimiser le temps d'exécution.

3.3.2.2 VIP

Développé par le laboratoire CREATIS, VIP (Virtual Imaging Platform) [45] est un portail ouvertement accessible pour faciliter le partage des données et l'accès aux ressources informatiques de la grille pour la simulation des images médicales. En utilisant le portail Web, les utilisateurs transfèrent des données et tournent des chaines de traitement d'applications sur la grille. Les applications de VIP sont exécutées sur l'organisation virtuelle Biomed de l'infrastructure de grille européenne EGI. L'ordonnancement de la plate-forme utilise des boucles d'autogestion [61] qui optimisent la finesse et l'équité dans l'exécution des chaines de traitement. Une des boucles contrôle la granularité de tâches tandis que l'autre contrôle l'équité de l'ordonnancement. Elles sont intégrées dans MOTEUR, le gestionnaire de workflow utilisé par la plate-forme VIP [62]. Les performances de ces boucles d'autogestion ont été évaluées dans des conditions réelles, sur l'infrastructure d'EGI. Les résultats obtenus montrent que la boucle de contrôle de la granularité de la tâche peut accélérer les exécutions jusqu'à un facteur 2 tandis que la boucle de contrôle de l'équité réduit la variabilité du ralentissement d'un facteur 3 à 7 par rapport à la politique FIFO. Les évaluations montrent aussi que la boucle de contrôle de la granularité des tâches dégrade l'équité, mais que la boucle de contrôle de l'équité compense cette dégradation.

3.3.2.3 Plates-formes à agents pilotes sur EGI

Plusieurs plates-formes à agents pilotes ont été proposées au cours des dernières années, telles que DIANE [63], WISDOM Production Environment [64], Panda [65], DIRAC [43] et glideInWMS [66]. La politique d'ordonnancement dans ces plates-formes à agents pilotes est conçue pour un usage général. Par exemple, le WISDOM Production Environment et DIANE utilisent la politique FIFO tandis que DIRAC utilise la politique quasi Round Robin. La plate-forme de DIRAC offre en plus la possibilité de configurer la politique d'ordonnancement pour

un groupe d'utilisateurs partageant une même application. Ainsi, nous pouvons appliquer une politique appropriée pour un groupe d'utilisateurs de criblage virtuel qui partagent le même logiciel de docking pour améliorer l'équité du système.

PANDA (Production And Distributed Analysis system, https://twiki.cern.ch/twiki/bin/view/PanDA/PanDA) est utilisé par la collaboration ATLAS pour gérer et distribuer ses calculs dans le monde entier [65]. PANDA utilise Condor-G (http://research.cs.wisc.edu/condor/description.html) comme système de soumission des agents pilotes sur les infrastructures (LHC, Open Science Grid). L'ordonnancement des tâches soumises à la plate-forme PANDA utilise un algorithme ad hoc développé pour les différents scénarii pertinents pour une grande expérience de physique des particules comme la production (génération, simulation et reconstruction de données) ou l'analyse de données. Les tâches de production et d'analyse ont des cahiers des charges différents et sont donc traitées différemment par l'ordonnanceur (voir https://twiki.cern.ch/twiki/bin/view/PanDA/PandaBrokerage).

3.3.3 Plate-forme HTCaaS

A la différence des plates-formes déployées sur EGI et/ou LCG décrites ci-dessus, la plate-forme HTCaaS déploie des agents privés sur des clusters et des supercalculateurs en Corée du Sud. L'ordonnancement consiste alors à s'assurer de l'équité entre plusieurs utilisateurs en termes de nombre d'agents. Pour adapter la répartition globale à la charge de l'utilisateur et aux ressources, HTCaaS a besoin d'un mécanisme de surveillance des changements globaux de distribution de charge et s'adapter efficacement au changement de charge. L'algorithme utilisé est un algorithme de Fair Share qui divise l'ensemble des ressources informatiques disponibles équitablement entre tous les utilisateurs exigeants dans le système. La fonction d'attribution des ressources RA (U) est définie selon l'équation 2.2 ci-dessous, où U représente un utilisateur dans le système.

$$RA(U) = \min\left(NumTasks(U), \frac{AvailableCores}{\sum_{p \in DU} Weight(p)} * Weight(U) \right) \quad (2.2)$$

Dans l'équation 2.2, les paramètres suivants sont introduits :

- NumTasks (U) est le nombre de tâches en attente de l'utilisateur U. Il est initialisé au nombre de tâches soumises par l'utilisateur U.
- AvailableCores est le nombre de ressources informatiques qui peuvent être acquises par les agents (y compris les processeurs libres et ceux qui sont déjà attribués) et nous supposons un mappage un-à-un entre un agent et un cœur de processeur dans le système.
- Weight(p) représente le poids d'un utilisateur p qui peut tenir compte de nombreux facteurs différents, tels que le nombre de tâches soumises par l'utilisateur p, le temps de calcul de tâche, des éléments de priorité, etc. Dans l'implémentation actuelle de HTCaaS, la fonction de poids pour tous les utilisateurs est égale à 1 (Weight(p) = 1), de sorte que $\Sigma_{p \epsilon DU}$Weight (p) est le nombre d'utilisateurs exigeants dans le système. La politique d'ordonnancement de HTCaaS est finalement de type Round Robin.

3.3.4 Ordonnancement sur grilles d'ordinateurs personnels

L'ordonnancement sur les grilles d'ordinateurs personnels (Desktop grids) présente de fortes spécificités. En effet, l'enjeu est de pallier de la façon la plus efficace possible à la perte de performance due à la volatilité des ressources, leur extrême hétérogénéité et l'absence de confiance et de fiabilité. La recherche sur le gestion des ressources et l'ordonnancement reste donc un sujet d'actualité sur les infrastructures gérées avec XtremWeb [79] ou Boinc [80]. SpeQuloS [81] propose ainsi un cadre pour améliorer la qualité de services des infrastructures distribuées de type « Best effort », incluant les infrastructures d'ordinateurs personnels.

3.4 Ordonnancement sur cloud

Nous avons vu comment l'utilisation de plates-formes à agents pilotes permettait de tirer le meilleur parti des ressources de la grille. Un des atouts de ces plates-formes est leur capacité à soumettre les agents pilotes sur d'autres ressources que la grille : clusters, clouds ou supercalculateurs.

3.4.1 Exemple de DIRAC

Par exemple, la portabilité de la plate-forme DIRAC sur un environnement de Cloud a été étudiée pour répondre aux besoins de l'expérience Belle sur accélérateur de particules au KEK (Japon). Une version Belle-DIRAC [67] a été conçue pour fonctionner sur Amazon EC2 Cloud pour augmenter l'efficacité opérationnelle et réduire les coûts. Après le succès de la version Belle-DIRAC, une extension Cloud de DIRAC (VMDIRAC) [68] a été développée pour permettre l'exécution des jobs pilotes sur les fédérations de clouds académiques (EGI, France-Grilles). VMDIRAC est utilisée pour instancier, surveiller et gérer des machines virtuelles dans plusieurs agrégations IaaS de Amazon EC2, OpenNebula, OpenStack et CloudStack.

Dans l'environnement du cloud, la plate-forme DIRAC utilise le même modèle « PULL » que dans la grille [69]. La figure 3.7 montre la comparaison entre DIRAC sur la grille et sur le cloud : au lieu de soumettre un job comme agent pilote sur la grille, la plate-forme demande une nouvelle machine virtuelle sur le cloud et télécharge l'agent pilote à cette machine.

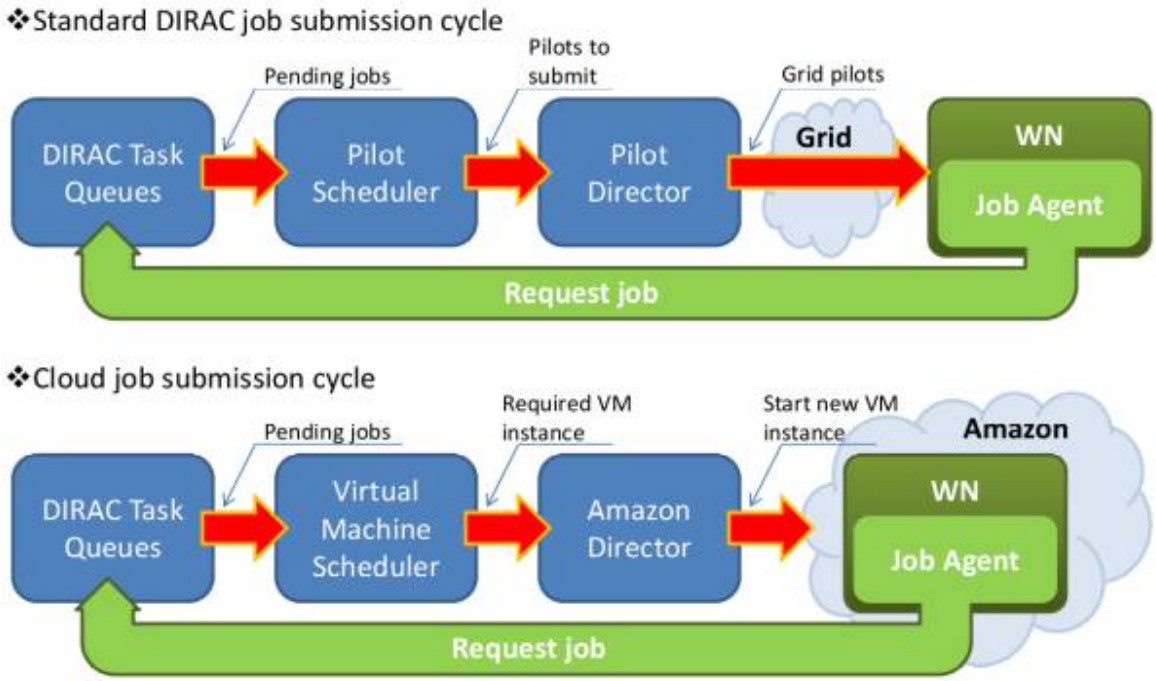


Figure 3-7: L'application du modèle d'agents pilotes (en haut) pour l'environnement du cloud (en bas) (Source: [69])

Nous retrouvons sur le cloud les deux niveaux d'ordonnancement préalablement identifiés sur la grille : l'ordonnancement au niveau de la plate-forme et l'ordonnancement au niveau du cloud lui-même.

Dans l'ordonnancement au niveau du Cloud, VMDIRAC gère les différentes Infrastructures-As-A-Service caractérisées par les paramètres suivants:

- Temps de réponse depuis l'envoi de la demande d'une machine virtuelle à l'attribution et à la contextualisation de la nouvelle machine virtuelle (Creation and Context Time)
- Vitesse de calcul des machines virtuelles
- Nombre maximum de machines virtuelles pouvant être créé par VMDIRAC sur l'IaaS

Dans l'ordonnancement au niveau de la plate-forme, la tâche de l'utilisateur est enregistrée dans la file d'attente de taches et l'algorithme d'ordonnancement est appliqué sur cette file d'attente. Dans l'environnement de la grille, les jobs ont la durée maximale de calcul autorisée sur un site donné tandis que les machines virtuelles sur le cloud fonctionnent jusqu'à ce que il n'y ait plus de tâche de l'utilisateur, puis s'arrête automatiquement et libére les ressources du Cloud. Ainsi, le problème de l'ordonnancement de la plate-forme sur le Cloud n'a pas de condition de disponibilité limitée de la machine. Par contre, l'ordonnancement doit tenir compte du coût si l'on considère le déploiement sur des clouds commerciaux. Dans le contexte de ce manuscrit, le coût n'a pas été pris en compte dans l'évaluation des politiques d'ordonnancement.

3.5 Conclusion

Dans ce chapitre, nous avons introduit le concept d'ordonnancement et présenté le contexte particulier de l'ordonnancement de plates-formes à agents pilotes sur grilles et clouds. Ces plates-formes connaissent un succès grandissant car elles réduisent la complexité d'accès à des ressources distribuées et cachent l'hétérogénéité des ressources à leurs utilisateurs.

Nous avons introduit la notion d'équité et expliqué l'intérêt d'optimiser deux paramètres caractéristiques des projets d'utilisateurs des plates-formes : le ralentissement moyen et le ralentissement maximum. Legrand et al. ont montré dans [55] que la politique SPT est très efficace pour optimiser ces paramètres.

Nous avons présenté les algorithmes utilisés sur les plates-formes couramment déployées sur les infrastructures de grille de production et en cours de déploiement sur les fédérations de clouds académiques en émergence, notamment FIFO, Round Robin et Fair Share. Aucune n'utilise la politique SPT à ce jour car celle-ci pénalise trop les utilisateurs avec beaucoup de

tâches, c’est-à-dire ceux qui déploient des analyses massives, communément appelées « data challenge ». Par rapport aux études de Legrand et al (3.2.4.3), le problème d'ordonnancement d’une plate-forme à agents pilotes sur grille utilise une définition centrée sur l'utilisateur et ajoute une contrainte supplémentaire: les machines ont une disponibilité limitée parce que la durée de vie des agents est limitée sur chaque centre de calcul de la grille. Par conséquent, le nombre de machines disponibles pour la plate-forme change avec le temps alors que les travaux de Legrand et al concernaient des machines dédiées constamment disponibles.

Nous avons aussi présenté le cas d’une plate-forme sur le cloud en prenant l’exemple de VMDIRAC.

Le travail qui va être exposé dans les chapitres suivants a consisté en une analyse comparative des performances des politiques d’ordonnancement utilisées sur les plates-formes (FIFO, RR, FS) aux politiques SPT et LPT en utilisant d’une part une modélisation dans SimGrid, d’autre part des expérimentations sur la grille EGI et sur un cluster au KISTI. Ce travail a débouché sur la proposition d’une nouvelle politique d’ordonnancement dont nous présenterons les performances obtenues par simulation et expérimentalement.

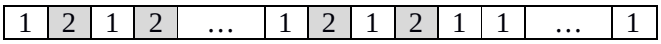
CHAPITRE 4 MODELISATION DU PROBLEME ETUDIE

Dans ce chapitre, nous allons poursuivre l'analyse de la problématique d'ordonnancement des plates-formes à agents pilotes. Après avoir illustré sur un exemple le ralentissement expérimenté par des utilisateurs, nous analyserons les caractéristiques des tâches de criblage virtuel pour démontrer qu'il est possible de préparer ces tâches de telle sorte qu'elles soient de durée approximativement constante, ce qui simplifie significativement la modélisation et l'expérimentation. Ensuite nous présenterons la formulation mathématique du problème d'ordonnancement et l'outil de modélisation SimGrid utilisé dans la suite.

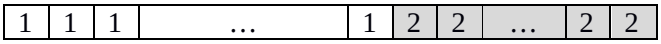
4.1 Définition du problème

Les algorithmes d'ordonnancement utilisés sur les plates-formes à agents pilotes sont des algorithmes simples comme FIFO (WPE, DIANE) ou quasi-RR (DIRAC). Le ralentissement des utilisateurs change avec la méthode d'ordonnancement alors que le temps de calcul pour toutes les tâches de tous les utilisateurs (makespan) n'est pas changé. Voici un exemple simple qui montre que la politique « temps de traitement le plus court d'abord » (SPT) fait mieux que FIFO et Round Robin pour optimiser le ralentissement des utilisateurs. Considérons deux utilisateurs : l'utilisateur 1 avec 20 tâches de docking, l'utilisateur 2 avec 10 tâches. Les deux utilisateurs soumettent des travaux à l'instant 0, mais l'utilisateur 1 arrive juste avant l'utilisateur 2. Pour simplifier, nous supposons que le temps d'exécution de toutes les tâches de docking est de 1 seconde. Observons l'ordre d'exécution des tâches selon les différents algorithmes dans les schémas ci-dessous:

- Avec Round Robin, l'utilisateur 1 se termine à t = 30s, l'utilisateur 2 à t = 20s



- Avec FIFO, l'utilisateur 1 a terminé à t = 20s, l'utilisateur 2 à t = 30s



- Avec SPT, l'utilisateur 1 a terminé à t = 30s, l'utilisateur 2 à t = 10s

2	2	…	2	2	1	1	1	…	1	1

Comme le résume le tableau 4.1, le temps total nécessaire pour exécuter toutes les tâches ne dépend pas de la politique d'ordonnancement (FIFO, SPT, Round Robin), mais le ralentissement des utilisateurs est différent entre les trois politiques. Dans cet exemple, FIFO et Round Robin génèrent de plus grands Sum-stretch (somme des ralentissements) et max-stretch (pire ralentissement) que SPT.

	Stretch utilisateur 1	Stretch utilisateur 2	Sum-stretch	Max-stretch	Temps total pour finir tous les jobs des utilisateurs (30 tâches)
FIFO	20/20=1	30/10=3	4	3	30
RR	30/20=1.5	20/10=2	3.5	2	30
SPT	30/20=1.5	10/10=1	2.5	1.5	30

Table 4-1: Stretch des utilisateurs

Il illustre bien le fait que l'ordonnancement des plates-formes à agents pilotes couramment utilisées sur EGI peut être amélioré.

Avant de proposer une formulation mathématique du problème étudié, nous nous sommes intéressés à la durée d'exécution des tâches de criblage virtuel sur la grille.

4.2 Analyse de la durée d'exécution des tâches de criblage virtuel

4.2.1 Description des jeux de données utilisés

Comme évoqué dans le chapitre 2, le criblage virtuel a pour objectif de calculer l'énergie de liaison d'un ligand au site actif d'une protéine d'intérêt dans le traitement d'une maladie. Nous avons choisi différents jeux de données pour analyser les propriétés des tâches de docking. Toutes nos analyses ont été menées avec le logiciel Autodock (Morris et al, 2009), qui fait partie des logiciels libres de référence pour le criblage virtuel. Développé au Scripps Research Institute (http://autodock.scripps.edu), il est distribué avec une suite d'outils graphiques pour préparer les expériences de criblage virtuel, ainsi que plusieurs tutoriels,

améliorant ainsi son accessibilité. Cité dans plusieurs articles scientifiques (Park, Lee et Lee, 2006)(Lin et al, 2002) (Lin et al, 2003), il a joué un rôle majeur dans le développement préclinique de la molécule Raltegravir (inhibiteur de l'intégrase du VIH), dont la distribution sur le marché américain a été autorisée fin 2007 par la FDA (Schames et al, 2004). Nous avons utilisé pour nos tests la version Autodock Vina (Trott & Olson, 2010) bénéficiant notamment d'une nouvelle fonction de score de l'ancrage entre ligand et cible.

La base de données fournie par l'INPC rassemble un total de 323 ligands de composés extraits à partir des plants vietnamiens. Au démarrage de la thèse, cette base de données n'était pas encore disponible et nous avons choisi pour nos tests d'utiliser la base de données ZINC (Irwin, Sterling, Mysinger, Bolstad, & Coleman, 2012) qui fournit gratuitement les structures de composés disponibles dans le commerce pour le criblage virtuel. ZINC contient plus de 35 millions de composés dans des formats compatibles avec les logiciels de docking les plus courants. ZINC est fourni par le Laboratoire Shoichet au département de chimie pharmaceutique à l'Université de Californie, San Francisco (UCSF).

Une fois la base de données de ligands et le logiciel de criblage choisis, le troisième élément indispensable est bien sur la cible biologique. Nous avons choisi en partenariat avec les chercheurs de l'INPC une protéine (identifiant PDB : 1OKE) de surface du virus de la dengue (Modis et al, 2003), une acétylcholinestérase (identifiant PDB : 1EVE), cible biologique de la maladie d'Alzeihmer déjà utilisée par le médicament ARICEPT (Kryger, Silman & Sussman, 1999) ainsi que la mortaline (identifiaint PDB : 3N8E, http://www.rcsb.org/pdb/explore/explore.do?structureId=3N8E), protéine d'intérêt pour le traitement de l'hépatocarcinome.

Nous avons effectué des tests de durée d'exécution des tâches de criblage sur un ordinateur personnel doté d'un processeur Intel Core Duo 2GHz. La paramétrisation du logiciel Autodock a été fournie par des chimistes de l'INPC.

4.2.2 Criblage virtuel sur une cible biologique de la dengue

La figure 4-1 présente un histogramme des temps d'exécution sur un ordinateur personnel du criblage de 2108 ligands extraits de la librairie ZINC contre la protéine 1OKE d'intérêt pour le traitement de la dengue avec le logiciel Autodock Vina. La plupart des temps de calcul sont

compris entre 12’ et 22’ (85,58%) avec un temps de calcul le plus court observé de 6'13’’ et le plus long de 29'14’’.

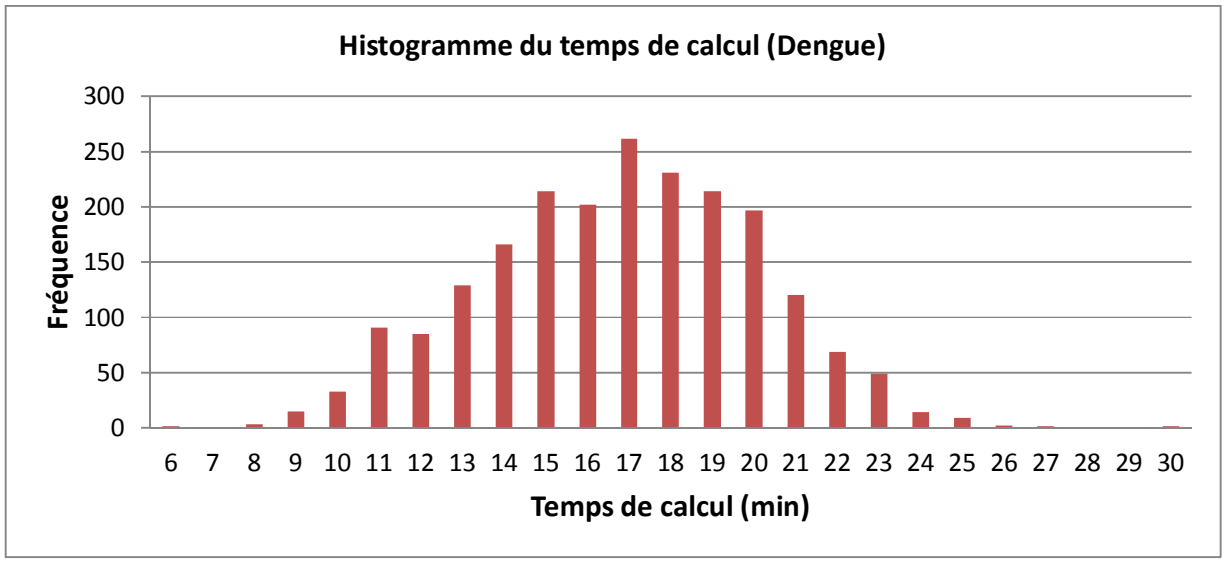


Figure 4-1: Histogramme du temps de calcul de docking contre la protéine 1OKE (virus de la dengue)

La figure 4-2 présente la variation du temps de calcul en fonction de la taille du fichier de ligand. Elle fait clairement apparaître une forte corrélation.

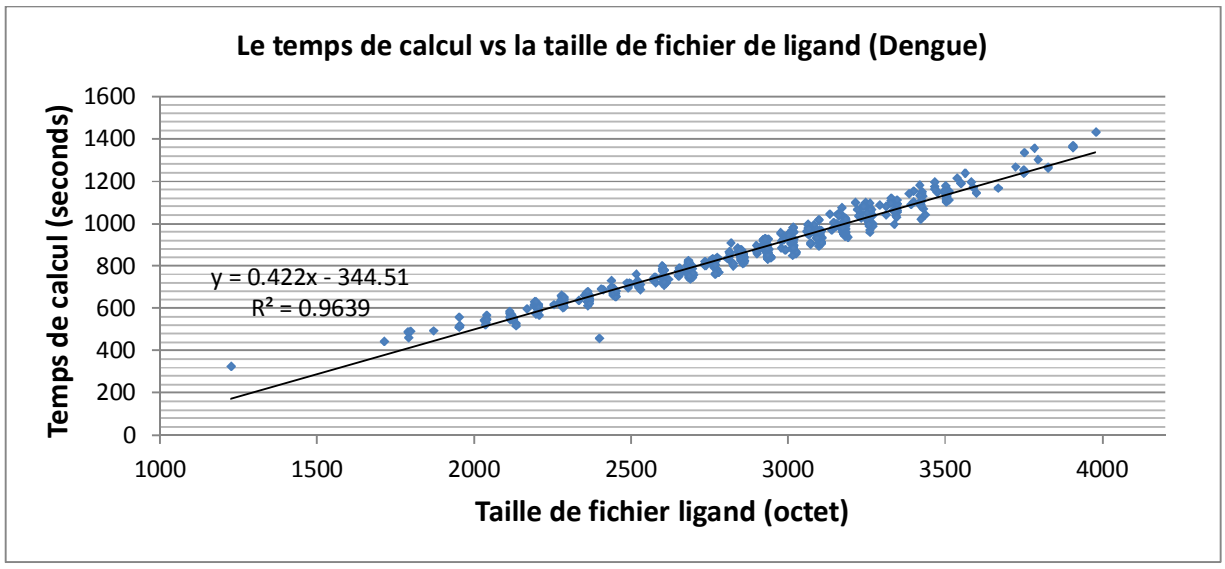


Figure 4-2: Relation entre la taille de fichier de ligand et le temps de calcul dans le projet de criblage virtuel contre la dengue

Le meilleur ajustement d’une corrélation linéaire entre la taille du fichier de ligand et le temps de calcul est représenté sur la figure 4-2:

$$y = 0.422x - 344,51$$

avec R^2 = 0,9639 (R^2 est le coefficient de détermination qui représente l'accord entre la fonction et la donnée réelle. La définition de R^2 est représentée dans [70].

4.2.3 Criblage virtuel sur une cible biologique de la maladie d'Alzheimer

La figure 4-3 présente la distribution des temps de calcul de criblage virtuel de 1392 ligands extraits de ZINC sur la protéine 1EVE d'intérêt pour le traitement de la maladie Alzeihmer. Les temps de calcul varient pour la plupart (95,85%) entre 4 et 12 minutes. Le temps de calcul le plus long observé est de 25'8'', le temps le plus court de 4'21''.

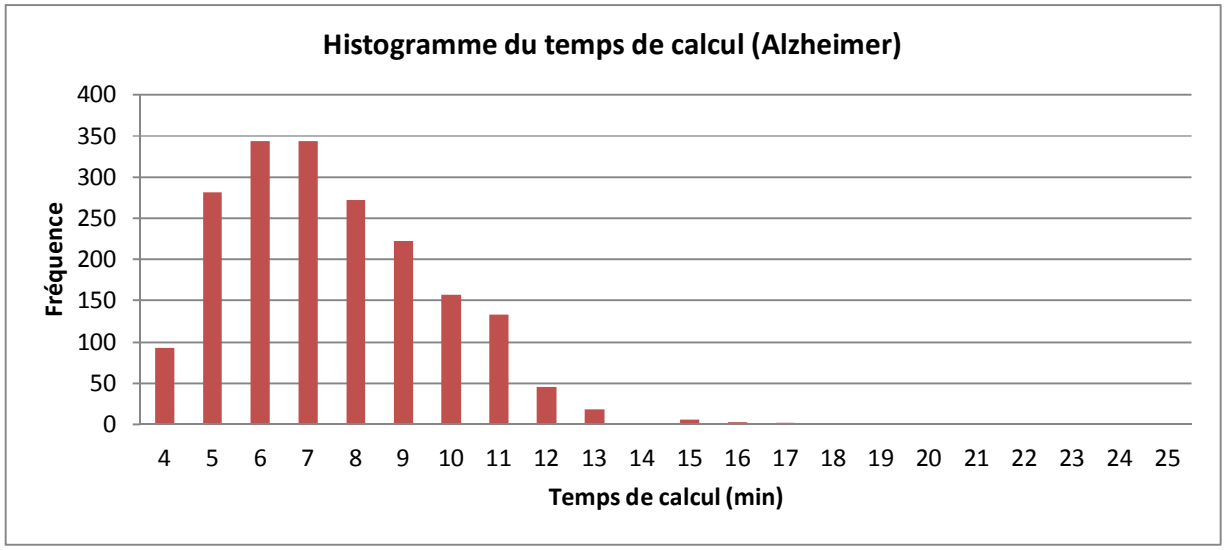


Figure 4-3 : Histogramme du temps de calcul de docking contre la protéine 1EVE (maladie Alzheimer)

La figure 4-4 présente la corrélation entre le temps de calcul et la taille du ligand. A nouveau, une corrélation linéaire est observée.

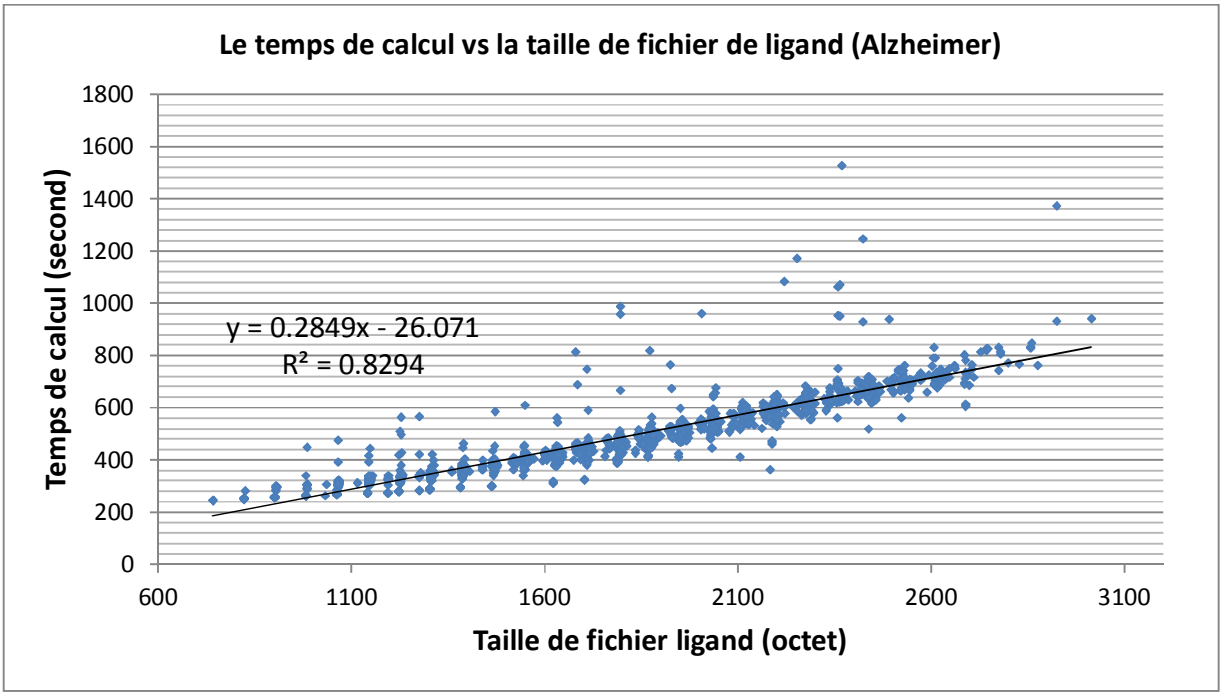


Figure 4-4 : Relation entre la taille de fichier de ligand et le temps de calcul dans le projet de criblage virtuel contre la maladie Alzheimer

Le meilleur ajustement de la relation linéaire entre la taille du fichier de ligand et le temps de calcul de docking est: $y = 0.2849x - 26.071$ avec $R^2 = 0.8294$

4.2.4 Criblage virtuel sur les données de l'INPC

La figure 4-5 présente la distribution des temps de calcul de criblage virtuel avec Autodock Vina de 323 ligands extraits de la base de données de l'INPC sur la protéine mortaline (3N8E) d'intérêt pour le traitement de l'hépatocarcinome. La plupart des temps de calcul se situent entre 4 et 10 minutes. Le temps de calcul le plus long observé est de 62', le plus court est de 4'12''.

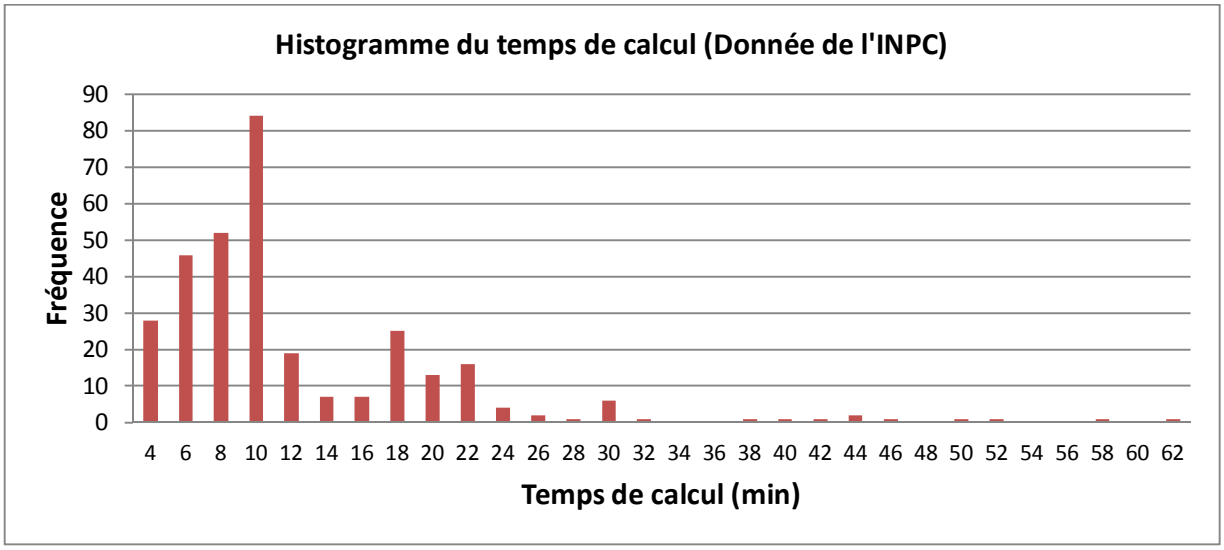


Figure 4-5: Histogramme du temps de calcul de docking (la donnée de l'INPC) contre la protéine 3N8E

La figure 4-6 présente les mêmes temps de calcul en fonction de la taille des fichiers.

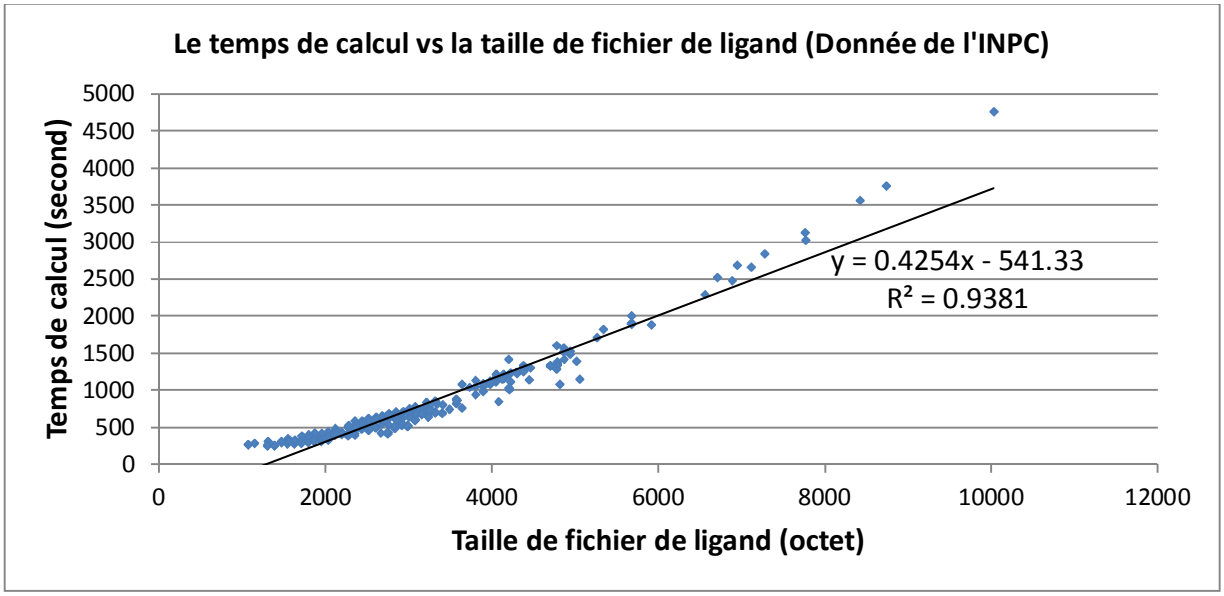


Figure 4-6: Relation entre la taille de fichier de ligand et le temps de calcul dans le projet criblage virtuel avec les données de l'INPC

Le meilleur ajustement linéaire de la relation entre la taille du fichier de ligand et le temps de calcul est présenté sur la figure 4-6:

$$y = 0.4254x - 541.33$$

avec $R^2 = 0.9381$

4.2.5 Conséquences sur le problème étudié

L'analyse de la durée des tâches individuelles de criblage virtuel d'un ligand sur une cible avec Autodock Vina met en évidence plusieurs comportements :

- la durée des tâches est homogène, de l'ordre de quelques minutes
- elle est très corrélée à la taille des ligands
- cette corrélation peut être approximée par une relation linéaire
- Le temps de calcul par octet de fichier de description de la structure tridimensionnelle des ligands varie d'une cible à l'autre

La durée des tâches individuelles étant de l'ordre de quelques minutes, nous regroupons les exécutions des criblages virtuels par lots. Il nous apparaît clairement que ce regroupement est pertinent pour l'optimisation des ressources de la grille car il évite une accumulation des temps de latence à cause d'un grand nombre de tâches de durée très courte. La connaissance préalable de la taille des fichiers de description de la structure tridimensionnelle des ligands permet d'organiser le regroupement des tâches de telle sorte que la durée d'exécution d'un lot d'exécution de criblage virtuel soit approximativement constante pour une machine donnée (Bin-Packing). Nous étudierons l'effet du regroupement des tâches dans la partie 4.4.4.2.

Fort de ce constat, nous avons choisi de conduire notre travail de modélisation et de simulation avec des tâches de durée constante. Nous allons à présent proposer une formulation mathématique complète du problème d'ordonnancement étudié.

4.3 Formulation

4.3.1 Caractéristiques des utilisateurs

4.3.1.1 Introduction

Une recherche sur la charge réelle de la grille [71], [72] a montré qu'il existe deux groupes d'utilisateurs de la grille avec différents comportements: un groupe d'utilisateurs ordinaires dits « normaux » et un groupe d'utilisateurs avec beaucoup de tâches baptisés « Data Challenge users». Les utilisateurs « normaux » se présentent selon une distribution régulière

et soumettent souvent un petit nombre de tâches sur la grille. Le groupe d'utilisateurs avec beaucoup de tâches « Data Challenge users » accèdent à la grille irrégulièrement et beaucoup moins fréquemment mais soumettent parfois un très grand nombre de tâches.

Nous retrouvons ces comportements parmi les utilisateurs du service DIRAC-FG, plate-forme DIRAC opérée par France Grilles et ouverte à plus de 15 organisations virtuelles (https://dirac.france-grilles.fr). La figure 4-7 montre le CPU consommé par les principaux utilisateurs du service depuis sa création. Parmi les 115 utilisateurs recensés, les 5 premiers utilisateurs représentent à eux seuls environ 80% des ressources consommées, les 10 premiers utilisateurs plus de 90%. La figure montre aussi que l'usage des ressources par ces utilisateurs n'est pas régulier, à l'exception du certificat robot du service VIP (utilisateur CREATIS, 2ème plus gros utilisateur). Les autres lancent des grosses productions, qui durent de quelques semaines à quelques mois. La figure illustre la difficulté de mettre en œuvre des politiques d'ordonnancement de type SPT (Shortest Processing Time) privilégiant systématiquement les utilisateurs normaux. En effet, les très utilisateurs avec beaucoup de tâches peuvent se retrouver pénalisés de façon systématique et c'est pour éviter cette situation qu'aucun plate-forme sur la grille n'applique ces politiques, préférant des politiques moins performantes (Round Robin ou FIFO) ou perçues comme plus équitables (Fair Share).

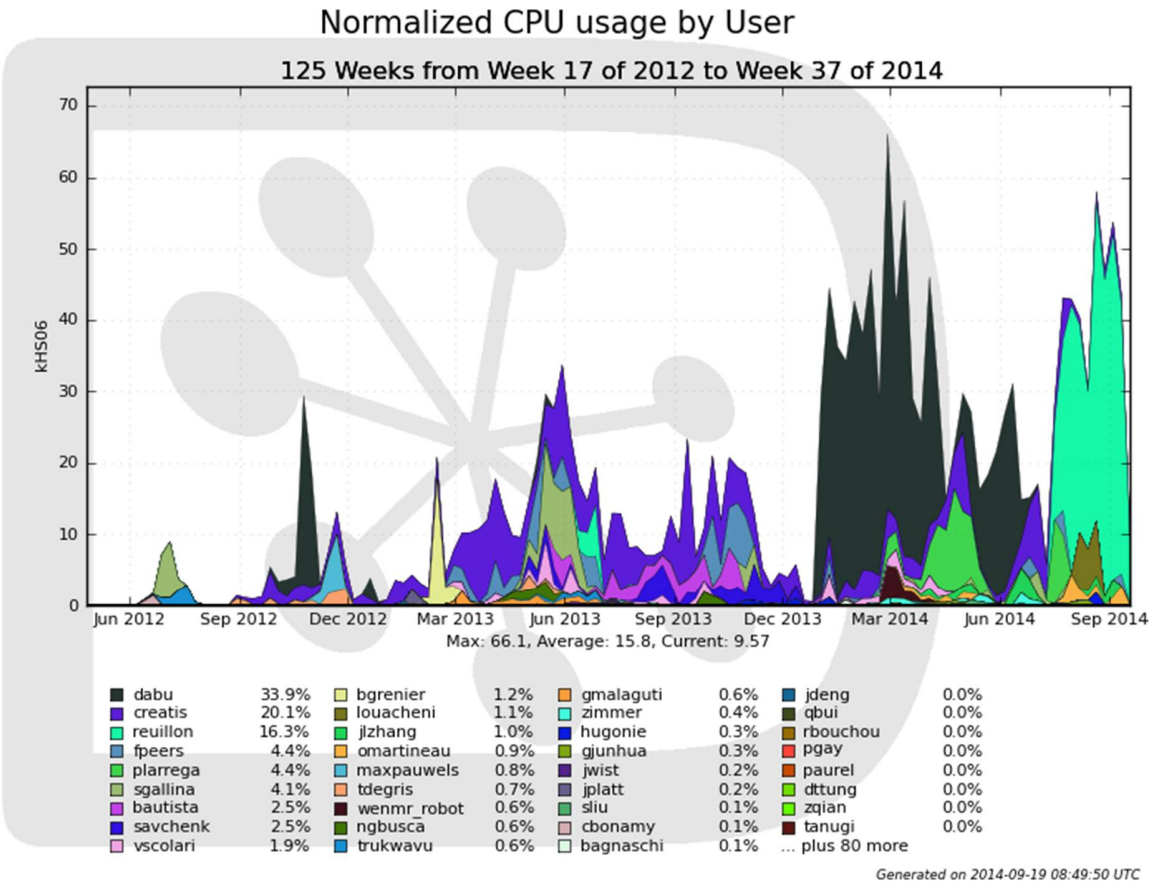


Figure 4-7 : Consommation des ressources (CPU exprimé en unité normalisée kHS06) par les utilisateurs du service DIRAC-FG de Juin 2012 à Septembre 2014 (source : https://dirac.france-grilles.fr)

Les comportements décrits sur la figure 4-7 sont attendus des utilisateurs d'une plate-forme pour le criblage virtuel. La majorité des utilisateurs lance des petits groupes de tâches pour optimiser une paramétrisation, tester des configurations ou des ligands d'action inhibitrice connue. Après ces tests, ils lancent des productions massives de données *in silico* dont l'analyse requiert ensuite des mois de travail.

4.3.1.2 Notations

Dans la communauté des utilisateurs du criblage virtuel, on considère X^{DC} utilisateurs avec beaucoup de tâches (« Data Challenge users ») et X^{Normal} utilisateurs normaux (« Normal users»). On dénote l'utilisateur i dans le groupe normal (pour i de 1 à X^{Normal}) et l'utilisateur j dans le groupe DC (Data Challenge) (pour j de 1 à X^{DC}).

L'utilisateur « Normal » i soumet un projet du criblage virtuel de tâches de criblage sur la grille/cloud et toutes les tâches du projet sont soumises à la plate-forme au temps r_i^{normal} .

L'utilisateur « Data Challenge » j soumet un projet du criblage virtuel de tâches de criblage sur la grille/cloud et toutes les tâches de projet j sont soumises à la plate-forme au temps r_j^{DC} .

Notation	Description
X^{DC}	Nombre des utilisateurs dans le groupe « Data Challenge »
X^{Normal}	Nombre des utilisateurs dans le groupe «Normal»
U_i^{normal}	Utilisateur i dans le groupe « Normal »
U_j^{DC}	Utilisateur j dans le groupe « Data Challenge »
N_i^{normal}	Nombre de tâches dans le projet d'utilisateur i dans le groupe « Normal »
N_j^{DC}	Nombre de tâches dans le projet d'utilisateur i dans le groupe « Data Challenge »
r_i^{normal}	Temps de soumission du projet d'utilisateur i dans le groupe « Normal »
r_j^{DC}	Temps de soumission du projet d'utilisateur i dans le groupe « Data Challenge »

Table 4-2: Notations caractéristiques de l'utilisateur

4.3.2 Caractéristiques de l'environnement de calcul

4.3.2.1 Caractéristiques de l'environnement de la grille

La grille est constituée de Y clusters. Notons C_j le cluster j (pour j variant de 1 à Y). Le cluster C_j est composé de NC_j machines qui ont la même vitesse relative α_j : par exemple, une tâche de criblage exécutée pendant P secondes sur un processeur de vitesse 1 va passer un temps de P/α_j (secondes) sur une machine dans le cluster C_j. En outre, chaque cluster C_j impose une limite D_j à la durée des jobs de la grille. Les jobs dont le temps de calcul est supérieur à D_j sont tués.

4.3.2.2 Caractéristiques des utilisateurs

Nous séparons les utilisateurs de la grille en deux groupes: les utilisateurs de criblage virtuel (VU) qui soumettent des projets de criblage virtuel à la plate-forme pilote-agent; et les utilisateurs de grille (GU) qui soumettent d'autres types de jobs.

A l'instant t, il y a $VU_{t(j)}$ machines dans le cluster C_j à exécuter des tâches des utilisateurs de criblage virtuel VU. Et il y a $GU_{t(j)}$ machines dans le cluster C_j à exécuter des tâches des utilisateurs normaux GU de la grille.

Les charges générées par un utilisateur de criblage virtuel à l'instant t, WL_VU_t, et pendant un laps de temps [t_begin, t_end], $WL\_VU_{[t\_begin,\ t\_end]}$, sont définies comme:

$$WL\_VU_t = \frac{\sum_{j=1}^{Y} \alpha_j . VU_{t(j)}}{\sum_{j=1}^{Y} \alpha_i . NC_i} \text{ et } WL\_VU_{[t\_begin, t\_end]} = \int_{t\_begin}^{t\_end} WL\_VU_t d_t$$

De même, les charges générées par un utilisateur normal à l'instant t, WL_GU_t et pendant une durée [t_begin, t_end], $WL\_GU_{[t\_begin,\ t\_end]}$, sont définies comme suit:

$$WL\_GU_t = \frac{\sum_{j=1}^{Y} \alpha_j . GU_{t(j)}}{\sum_{j=1}^{Y} \alpha_i . NC_i} \text{ et } WL\_GU_{[t\_begin, t\_end]} = \int_{t\_begin}^{t\_end} WL\_GU_t d_t$$

4.3.2.3 Caractéristiques des agents pilotes

Il y a Z agents pilotes dans le système : ce nombre varie en fonction du temps. L'agent pilote est denoté PA_s (pour s = 1 ... Z). L'agent pilote PA_s s'exécute sur une machine du cluster $C_{j(s)}$ pour la durée de temps maximale $D_{j(s)}$. L'agent pilote s'exécute dans l'intervalle de temps [B_s, E_s]. La connexion entre l'agent pilote PA_s et sa plate-forme présente une latence $L_{j(s)}$.

Le temps d'exécution d'une tâche de criblage sur n'importe quelle machine du cluster C_j est égal à P/α_j. Nous définissons le temps perdu:

$$\Delta = D_{j(s)} - P*(D_{j(s)} \text{ modulo } P)$$

Pendant le temps Δ, la machine est inutilisée parce que l'agent pilote n'a pas assez de temps pour exécuter une nouvelle tâche.

4.3.2.4 Caractéristiques des tâches des utilisateurs

L'utilisateur Ui a N_i tâches à exécuter. La $k^{ème}$ tâche de l'utilisateur U_i est exécutée par $PA_{s(k)}$ et commence à l'instant T_k. Une seule tâche est exécutée à un moment donné par un agent pilote. Nous avons donc les contraintes suivantes:

$$T_k \geq B_{s(k)} \text{ et } T_k + P_k/\alpha_{j(s(k))} \leq E_{s(k)}.$$

4.3.2.5 Caractéristiques de la plate-forme à agents pilotes sur la grille

L'algorithme de contrôle de la soumission des agents dans le module de gestion des agents (Agent Manager) de la plate-forme est le suivant:

Algorithm 4.1: Algorithme de contrôle de la soumission des agents pilotes

```
start Task Manager
submit init pilots
WHILE Task Manager is alive DO
 sleep (iteration time)
 n = number of waiting tasks – number of waiting agent
 sub = min (maxSub,n) pilots
 submit sub pilots
END WHILE
```

La plate-forme met les demandes des utilisateurs dans une file d'attente de tâches d'utilisateur. L'ordonnanceur dans le gestionnaire de tâches calcule la priorité de la tâche en fonction de la politique d'ordonnancement sélectionnée. Le but de notre recherche est d'étudier et d'évaluer différentes politiques d'ordonnancement et de trouver une politique qui optimise le ralentissement (stretch) des utilisateurs.

Notation	Description
Y	Nombre des clusters dans la grille
P	Temps de calcul d'une tâche
C_j	Cluster j
NC_j	Nombre de machines dans le cluster j
α_j	La vitesse relative des machines dans le cluster j
D_j	La limitation de temps de calcul sur le cluster j
$VU_{t(j)}$	Nombre des machines dans le cluster j qui exécutent des tâches de criblage virtuel au temps t
$GU_{t(j)}$	Nombre des machines dans le cluster j qui exécutent des tâches normaux au temps t
WL_VU_t	La charge des utilisateurs de criblage virtuel au temps t
WL_GU_t	La charge des utilisateurs de grille au temps t
Z	Nombre d'agents pilotes
PA_s	Agent pilote s
$C_{j(s)}$	Cluster où l'agent pilote s'exécute
$D_{j(s)}$	Le temps maximal de calcul du cluster où l'agent pilote s'exécute
$[B_s, E_s]$	L'intervalle où l'agente pilote s exécute
L_s	La latence entre l'agente pilote s et la plate-forme
Δ	Temps perdu (défini par la formule du texte)

Table 4-3: Résume les notations utilisées.

4.3.2.6 Caractéristiques de l'environnement du cloud

La fédération de clouds considérée est constituée de Y clouds, appelés aussi « endpoints ». Notons E_j le endpoint j (j variant de 1 à Y). La plate-forme à agents pilotes peut créer un nombre maximal NC_j de machines virtuelles sur le endpoint E_j et ces machines ont la même vitesse relative α_j : par exemple, une tâche exécutée pendant P secondes sur un processeur de vitesse 1, passera un temps P/α_j secondes sur une machine dans le endpoint E_j. Le temps de la création de machine virtuelle du endpoint E_j est L_j.

Lorsqu'une machine virtuelle est créée, elle contacte le gestionnaire de tâche de la plate-forme pour télécharger les tâches à exécuter. S'il n'y a plus de tâche dans la file d'attente, la machine virtuelle va rester active pour une durée définie par le paramètre TIME_OUT. Après cette période, sans arrivée de nouvelle tâche, la machine virtuelle s'arrête.

4.3.2.7 Caractéristiques de la plate-forme à agents pilotes sur le cloud

L'algorithme de contrôle de la création de machine virtuelle de la plate-forme à agents pilotes sur le cloud utilise les paramètres CPUperInstance et Iteration_time. Le paramètre CPUperInstance définit la valeur seuil du temps de calcul des tâches dans la file d'attente pour démarrer une nouvelle machine virtuelle. L'ordonnanceur met à jour périodiquement l'état de la file d'attente après une période Iteration_time pour décider de soumettre ou pas une nouvelle machine virtuelle. L'algorithme de contrôle de la création de machine virtuelle de la plate-forme s'écrit ainsi (algorithme 4.2):

Algorithm 4.2: Algorithme de contrôle de la création de la machine virtuelle

```
start Task Manager
WHILE (Task Manager is alive)
DO
 n = number of waiting tasks in task queue of TM
 IF (n > CPUperInstance) THEN
 IF (end_point is available) THEN
  send request 1 VM to end_point
 ENDIF
 ENDIF
 sleep (iteration_time)
END WHILE
```

Cet algorithme simplifié ne prend pas en compte le type de machine virtuelle, ni le coût. Les demandes des utilisateurs sont mises dans une file d'attente. L'ordonnanceur du gestionnaire de tâches calcule la priorité de la tâche en fonction de la politique d'ordonnancement sélectionnée.

Notation	Description
Y	Nombre de clouds (« end-points ») fédérés
E_j	J^{ème} « end-point »
L_j	temps pour créer une nouvelle machine virtuelle
NC_j	Nombre maximal de machines virtuelles que la plate-forme peut créer simultanément

Table 4-4: Notation de caractéristiques de l'environnement du cloud

4.3.3 Fonction objectif

Le projet de l'utilisateur normal U_i^{normal} avec N_i^{normal} tâches est soumis à la plate-forme à l'instant r_i^{normal} . La k^{ème} (k $\in$ [1, N_i^{normal}]) tâche se termine au temps t_i^k .

E_i^{normal} est le temps d'achèvement de la dernière tâche dans le projet de l'utilisateur normal i:

$$E_i^{nomal} = \max_{k=1..N_i^{normal}} t_i^k .$$

Le ralentissement de l'utilisateur normal i est: $S_i^{normal} = \frac{E_i^{normal} - r_i^{normal}}{P * N_i^{normal}}$.

Max-stretch et moyen-stretch [$S_{\max}^{normal}$, $S_{average}^{normal}$] sont utilisées comme fonctions objectif à optimiser:

$$S_{\max}^{normal} = \max_{i \in [1..X^{normal}]} S_i^{normal} \text{ et } S_{average}^{normal} = \frac{\sum_{i=1}^{X^{normal}} S_i^{normal}}{X^{normal}}$$

Le projet de criblage virtuel de l'utilisateur Data Challenge U_i^{DC} avec N_i^{DC} tâches de criblage est soumis à la plate-forme à l'instant r_i^{DC} . La k-ième (k $\in$ [1, N_i^{DC}]) tâche de criblage est terminée au temps t_i^k . E_i^{DC} est le temps d'achèvement de la dernière tâche dans le projet de l'utilisateur Data Challenge i: $E_i^{DC} = \max_{k=1..N_i^{DC}} t_i^k$. Le ralentissement de l'utilisateur Data Challenge i est: $S_i^{DC} = \frac{E_i^{DC} - r_i^{DC}}{P * N_i^{DC}}$. Les paramètres max-stretch et moyen-stretch [$S_{\max}^{DC}$, $S_{average}^{DC}$] sont utilisés comme fonctions objectif:

$$S_{\max}^{DC} = \max_{i \in [1..X^{DC}]} S_i^{DC} \text{ et } S_{average}^{DC} = \frac{\sum_{i=1}^{X^{DC}} S_i^{DC}}{X^{DC}}$$

Il est très important de noter que, du fait des définitions adoptées, les ralentissements peuvent être inférieurs à 1. En effet, l'intérêt de la grille est précisément l'exécution des tâches en parallèle.

4.3.4 Description du problème dans le système de notation de Graham

Le problème d'ordonnancement peut être décrit dans le système de notation de Graham (Graham, Lawler, Lenstra, & Kan, 1979) :

Dans l'environnement de la grille, notre problème s'écrit de la façon suivante :

$$Q, NC_{win} \mid UET \mid [S_{\max}^{DC}, S_{average}^{DC}], [S_{\max}^{normal}, S_{average}^{normal}]$$

Et dans l'environnement du cloud :

$$Q \mid UET \mid [S_{\max}^{DC}, S_{average}^{DC}], [S_{\max}^{normal}, S_{average}^{normal}]$$

où :

- Q représente l'ensemble des machines dans l'infrastructure qui ont des vitesses différentes.
- NC_{win} indique que la disponibilité des machines ne suit pas un motif déterminé à l'avance. Cela signifie que la disponibilité des machines n'est pas connue à l'avance. Dans l'environnement du cloud, cette contrainte est levée.
- UET: comme nous l'avons expliqué précédemment, les projets de criblage virtuel peuvent être ramenés à l'exécution de tâches de taille unitaire, ce qui se traduit par la propriété « Unit Execution Time»
- $[S_{\max}^{DC}, S_{average}^{DC}]$ et $[S_{\max}^{normal}, S_{average}^{normal}]$ sont respectivement les ralentissements maximaux et la moyens des groupes « Data Challenge users» et « Normal users »

4.4 SIMGRID

Maintenant que nous avons défini précisément le problème étudié, nous allons clore ce chapitre en présentant le simulateur SIMGRID utilisé pour tester et évaluer les différentes politiques d'ordonnancement proposées sur la grille et le cloud.

4.4.1 Principes de fonctionnement de SIMGRID

SIMGRID (Casanova, Legrand, & Quinson, 2008) est un outil scientifique permettant d'étudier le comportement des systèmes distribués à grande échelle tels que les grilles, le Cloud/HPC ou des systèmes de P2P. Il fournit des modèles prêts à être utilisés et des API pour simuler de nombreux systèmes distribués: clusters, réseaux locaux et étendus, les centres de données, etc. La communauté des utilisateurs SIMGRID rassemble des centaines de membres enthousiastes dans le monde entier.

Des APIs visibles de SIMGRID pour l'utilisateur sont les suivantes (cf. figure 4.8) :

- SimDAG : comparaison d'ordonnancements heuristiques pour DAG
- GRAS : développement et débogage d'application réelle
- MSG: comparaison de méthodes heuristiques pour les processus séquentiels simultanés (CSP : Communicating Sequential Processes)
- SMPI : exécution d'applications MPI au- dessus de SIMGRID

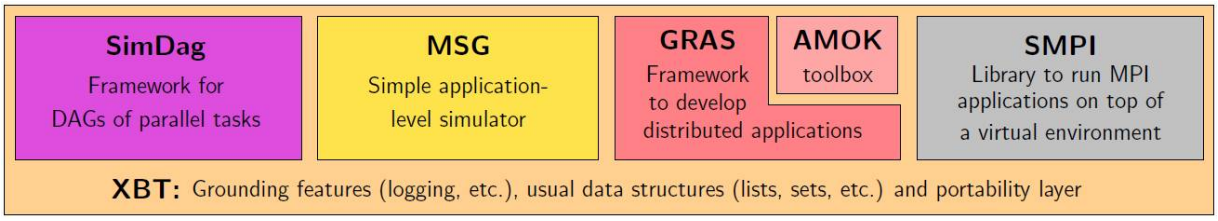


Figure 4-8: API visibles pour les utilisateurs

SIMGRID est écrit en C pour la vitesse et la portabilité. Tous les composants sont naturellement utilisables en C. Cependant, SIMGRID permet aux développeurs de construire le simulateur en Java ou C++. Notre simulateur est construit par composant MSG dans le langage C.

4.4.2 Structure du simulateur

4.4.2.1 Simulateur pour la grille

Sur une infrastructure de grille, il y a beaucoup d'utilisateurs avec des buts différents. Afin de simuler le fonctionnement d'une plate-forme d'agents pilotes sur une infrastructure de grille, nous devons simuler à la fois l'environnement de la grille et la plate-forme de criblage virtuel (figure 4-9). En conséquence, notre simulateur comporte deux modules principaux: un qui simule l'environnement de la grille et un autre qui simule la plate-forme d'agents pilotes.

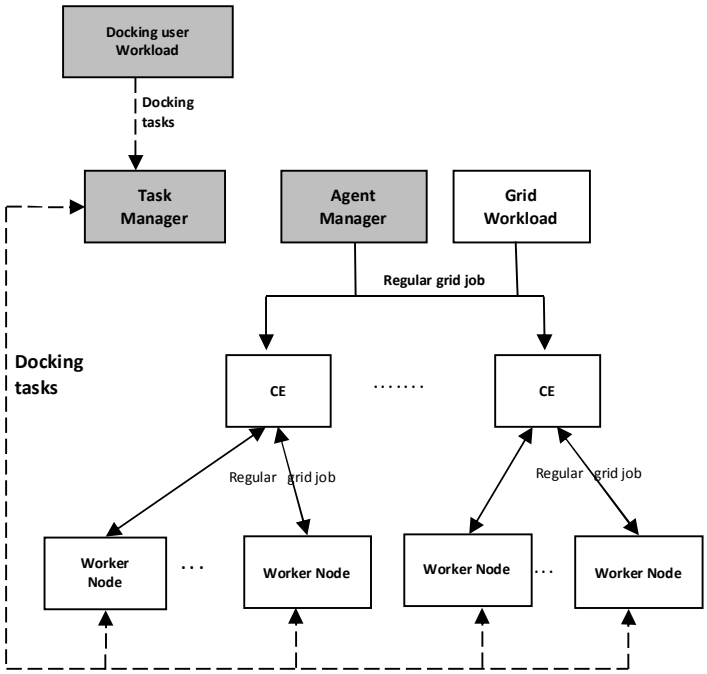


Figure 4-9: Structure du simulateur

L'infrastructure de grille (figure 4-10) est composée de nombreux centres de calcul, appelés «sites». Dans chaque site, l'élément de calcul (CE) contrôle les ressources de plusieurs worker node (WN): la disponibilité du processeur, la RAM, la capacité du disque dur, ...

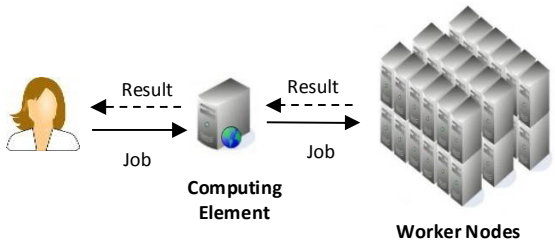


Figure 4-10: Modèle d'infrastructure de grille

Pour simuler l'environnement de grille, nous avons construit les modules suivants: **Grid Workload**, **Computing Element (CE)** et **Worker Node (WN)**.

Le module **Grid Workload** simule les demandes de l'utilisateur au CE. Ce module utilise l'archive de la charge de grille (Grid Workload Archive – GWA) comme donnée d'entrée pour générer les demandes des utilisateurs au CE. GWA contient des informations sur l'identité des tâches, l'heure d'arrivée, le nom du CE soumis, le temps de calcul dans le CE. Dans notre simulation, nous avons utilisé les archives de charges réelles sur l'infrastructure régionale AuverGrid en Auvergne pour la période 2004-2005.

Algorithme 4.2 : Contrôle de la tâche d'utilisateur et de l'agent pilote

```
J_Q : file d'attente de tâche
WN_Q : file d'attente du worker node
start CE
WHILE (CE is alive)
DO
 IF (received request from pilot agent) THEN
  IF (J_Q is not empty) THEN
   pop job from J_Q
   send job to WN
  ELSE
   put WN to WN_Q
  END IF
 END IF

 IF (received job from user) THEN
  IF (WN_Q is not empty) THEN
   pop worker node from A_Q
   send job to worker node
  ELSE
   put user job to J_Q
  END IF
 END IF
END WHILE
```

Le nœud CE gère deux files d'attente: la file d'attente des jobs et celle des WN (cf algorithme 4.1). Quand un CE reçoit un job de l'utilisateur, il cherche dans la file d'attente des WN. S'il y a un WN en file d'attente, le CE envoie le job à ce WN. Si non, le job de l'utilisateur est mis dans la file d'attente des jobs. Le WN envoie la demande de job à exécuter au CE quand il est

prêt à exécuter des nouveaux jobs. S'il y a une tâche dans la file d'attente des tâches, le CE envoie ce job à WN. Si non, le CE met le WN dans la file d'attente des WNs.

La simulation de la plate-forme comporte trois modules: **Agent Manager, Task Manager** et **Docking Workload**. Le module **Agent Manager** envoie automatiquement des agents pilotes au CE. Le CE trouve les WN disponibles et envoie des agents pilotes à ces WN. Un agent pilote en cours d'exécution sur un WN retire une tâche de docking à partir du module **Task Manager** pour l'exécuter. Lorsque le temps de calcul d'un agent pilote atteint la durée maximale de calcul allouée par le CE, l'agent pilote est automatiquement supprimé.

Le module **Docking Workload** envoie les tâches des utilisateurs au **Task Manager**. La file d'attente de tâche dans le module **Task Manager** gère les tâches de l'utilisateur et calcule les priorités conformément à la politique de l'ordonnancement. Lorsque le **Task Manager** reçoit une demande à partir d'un agent pilote, la tâche de l'utilisateur avec la plus haute priorité est envoyée à cet agent.

4.4.2.2 Simulateur pour le cloud

Nous avons modélisé le fonctionnement de VM-DIRAC sur le cloud fédéré d'EGI (EGI Federated Cloud - http://www.egi.eu/infrastructure/cloud/index.html) qui est formé par les clouds des instituts membres de l'EGI. La plate-forme VM-DIRAC est en cours de connexion à ces clouds. Chaque cloud est considéré comme un « end-point » (figure 4-11) qui va fournir des machines virtuelles.

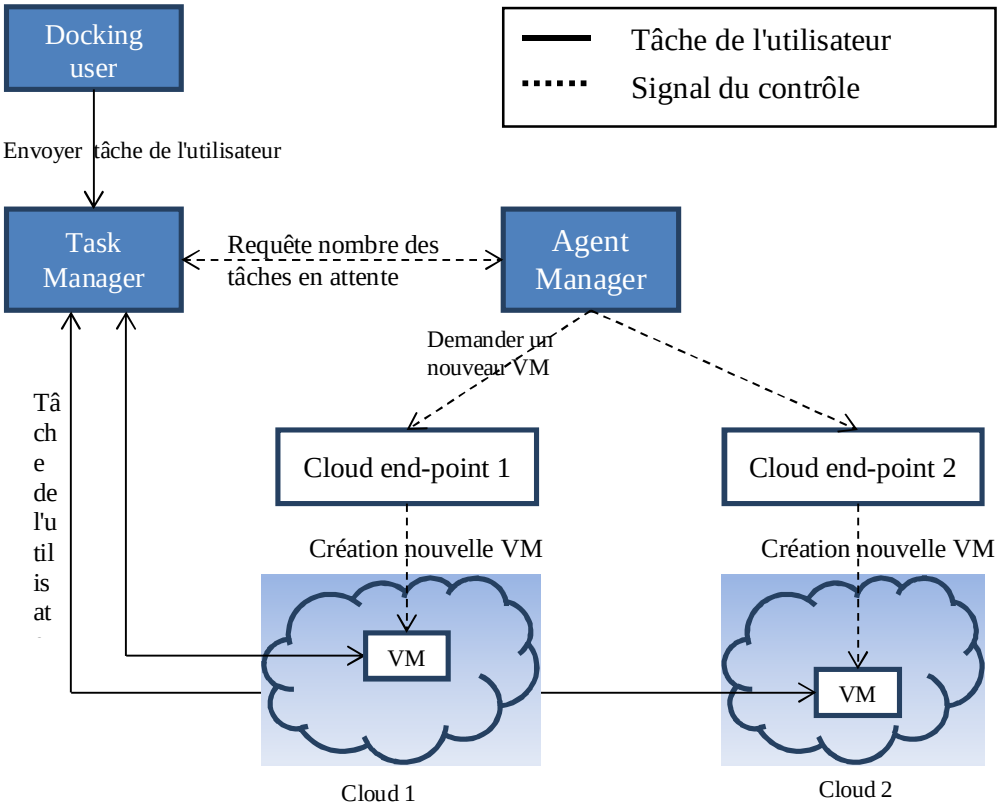


Figure 4-11: Structure du simulateur pour la plate-forme à agents pilotes sur le cloud

Quand il y a des tâches dans la file d'attente du gestionnaire de tâches (Task Manager), le gestionnaire d'agents (Agent Manager) envoie une requête à un end-point pour créer une nouvelle machine virtuelle (VM). Après sa création, la machine virtuelle est configurée et télécharge un agent pilote de la plate-forme. L'agent pilote en cours d'exécution sur la machine virtuelle va contacter le Task Manager pour prendre et exécuter des tâches de l'utilisateur.

Le simulateur de la plate-forme sur le cloud contient donc deux composantes principales, la plate-forme (Task Manager, Agent Manager et Docking Workload) et l'infrastructure de cloud (End-point et Virtual Machine).

Tout d'abord, le module « Docking User » envoie des tâches de docking au module « Task Manager », tâches qui sont mises dans sa file d'attente. Le module « Agent Manager » envoie périodiquement la requête sur le nombre de tâches en attente dans la file d'attente. Si le nombre de tâches de docking est supérieur à CPUperInstance, l'Agent Manager envoie alors une requête pour créer une nouvelle machine virtuelle au module End-point (cf algorithme

4.2). Sur la machine virtuelle ainsi créée, un agent pilote est démarré et commence à télécharger des tâches de l'utilisateur à exécuter.

4.4.3 Implémentation des politiques d'ordonnancement

Pour mieux comprend l'ordonnancement dans la plate-forme d'agents pilotes, nous présentons d'abord le mécanisme d'appariement (Matcher) dans le module « Task Manager » qui associe les tâches d'utilisateur avec les agents pilotes. Rappelons qu'il y a deux files d'attente dans le module « Task Manager », une file d'attente des jobs utilisateurs et une autre de nœuds de calcul (Worker Nodes) comme le montre la figure 4-12.

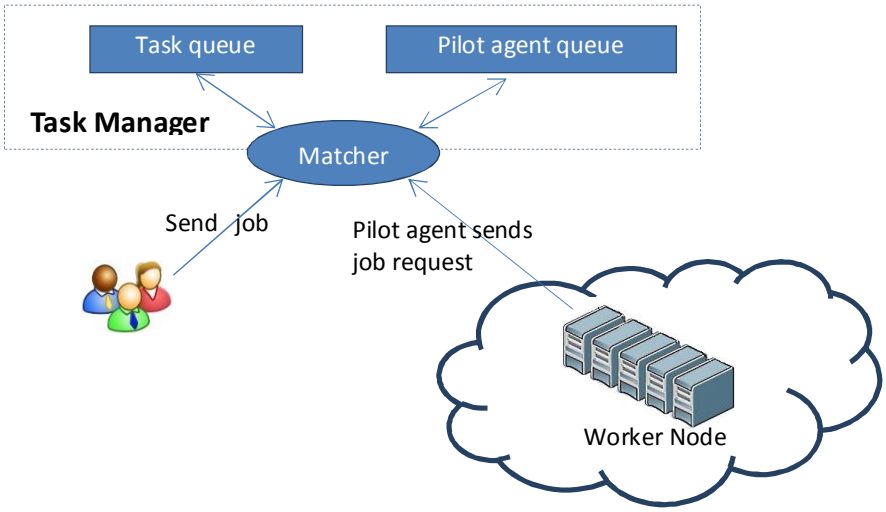


Figure 4-12: Deux files d'attente dans le module « Task Manager »

Algorithme 4.3 : L'algorithme d'appariement (Matcher) dans le module « Task Manager »

```
WHILE (Task Manager is alive)
DO
 IF (Receive request of pilot agent) THEN
 IF (there is job in Task Queue) THEN
  Get the first task in queue and send to pilot agent
 ELSE
  Put pilot agent in pilot agent queue
 ENDIF
 ENDIF

 IF (Receive task from User) THEN
 IF (there is pilot agent in pilot agent queue) THEN
  Get the first pilot agent in queue and send task to it
 ELSE
  Put task in the task queue
 ENDIF
 ENDIF
END WHILE
```

Sur la plate-forme à agents pilotes, des utilisateurs soumettent leurs tâches au module « Task Manager ». Ces tâches sont mises dans la file d'attente de tâches et la politique de l'ordonnancement (FIFO, SPT, LPT, RR, FS) est appliquée pour calculer leur priorité. Quand un agent pilote envoie une requête de tâche au module « Task Manager », celui-ci envoie à cet agent la tâche qui a la plus haute priorité dans la file d'attente.

4.4.4 Validation

Afin de valider le simulateur, nous allons évaluer l'impact du regroupement des tâches de docking en paquets sur le ralentissement. Nous avons d'abord effectué le calcul théorique de cet impact, et ensuite comparé avec le résultat obtenu à l'aide du simulateur.

4.4.4.1 Calcul théorique de l'impact du nombre de taches de docking dans un paquet sur le ralentissement

Considérons un utilisateur normal U_i^{normal} qui soumet un projet de criblage virtuel j avec $N_{j(i)}^{normal}$ tâches de docking (la même chose est vraie pour un utilisateur Data Challenge). Chaque tâche de docking requiert le même temps de calcul P. Lorsque la plate-forme à agents pilotes reçoit une demande d'un agent pilote disponible, elle regroupe x tâches de docking en

un paquet qu'elle envoie à l'agent pilote. Le nombre de paquets est: [$N_{j(i)}^{normal}$ / x] et le temps nécessaire à l'exécution d'un paquet de calcul est [P * x].

Supposons que le projet de criblage virtuel est composé de M agents pilotes déployés sur les machines de cluster C_j de même vitesse α_j égale à 1. Le temps de latence $L_{j(s)}$ entre les agents pilotes PA_s et la plate-forme est supposé constant L pour tous les agents pilotes. Le ralentissement expérimenté par un utilisateur normal i dans le projet j, est égal à

$$S_{j(i)}^{normal} = \frac{\left\{ \frac{N_{j(i)}^{normal}}{x.M} \right\} \times (P.x + L)}{N_{j(i)}^{normal}.P}$$

- Si $\frac{N_{j(i)}^{normal}}{x} < M$ (ou bien $x > \frac{N_{j(i)}^{normal}}{M}$), alors $\left\{ \frac{N_{j(i)}^{normal}}{M.x} \right\} = 1$

$$S_{j(i)}^{normal} = \frac{P.x + L}{N_{j(i)}^{normal}.P} = \frac{1}{N_{j(i)}^{normal}} x + \frac{L}{N_{j(i)}^{normal}.P} \qquad (4.1)$$

  Le ralentissement augmente linéairement par rapport à x quand $x > \frac{N_{j(i)}^{normal}}{M}$

- Si $\frac{N_{j(i)}^{normal}}{x} > M$ (ou bien $x < \frac{N_{j(i)}^{normal}}{M}$): $\left\{ \frac{N_{j(i)}^{normal}}{M.x} \right\} = \frac{N_{j(i)}^{normal}}{M.x} + \Delta$ $(0 \le \Delta < 1)$

$$S_{j(i)}^{normal} = \frac{\left(\frac{N_{j(i)}^{normal}}{x.M} + \Delta \right).(P.x + L)}{N_{j(i)}^{normal}.P} = \frac{\frac{N_{j(i)}^{normal}}{x.M}(P.x + L)}{N_{j(i)}^{normal}.P} + \Delta \frac{P.x + L}{N_{j(i)}^{normal}.P}$$

$$= \frac{1}{M} + \frac{L}{P.M} \times \frac{1}{x} + \Delta \left(\frac{x}{N_{j(i)}^{normal}} + \frac{L}{N_{j(i)}^{normal}.P} \right)$$

On note $\varepsilon = \frac{1}{M} + \Delta \left(\frac{x}{N_{j(i)}^{normal}} + \frac{L}{N_{j(i)}^{normal}.P} \right)$

Parce que $\frac{N_{j(i)}^{normal}}{x} > M \Rightarrow \frac{x}{N_{j(i)}^{normal}} < \frac{1}{M}$ et $0 \le \Delta < 1$, alors

$$\frac{1}{M} \le \varepsilon = \frac{1}{M} + \Delta\left(\frac{x}{N_{j(i)}^{normal}} + \frac{L}{N_{j(i)}^{normal}.P}\right) < \frac{2}{M} + \frac{L}{N_{j(i)}^{normal}.P}$$

On a:

$$S_{j(i)}^{normal} = \frac{L}{P.M} \times \frac{1}{x} + \varepsilon \quad \left(\text{avec } \frac{1}{M} \le \varepsilon < \frac{2}{M} + \frac{L}{N_{j(i)}^{normal}.P}\right) \qquad (4.3)$$

Le ralentissement augmente lorsque le nombre de tâches de docking dans un paquet diminue.

4.4.4.2 Résultat de test sur le simulateur de l'impact du nombre de taches de docking dans un paquet sur le stretch

Nous présentons sur la Figure 4-13 le résultat de simulation pour un utilisateur qui soumet un projet de 120.000 tâches. Le nombre de tâches par paquet prend les valeurs suivantes : 1, 2, 3, 4, 6, 12, 15, 20, 24, 30, 40, 60, 120, 200, 250, 300, 350, 400, 450, 500, 600, 700, 800 et 1000. Son projet est soumis sur une grille régionale avec environs 400 machines.

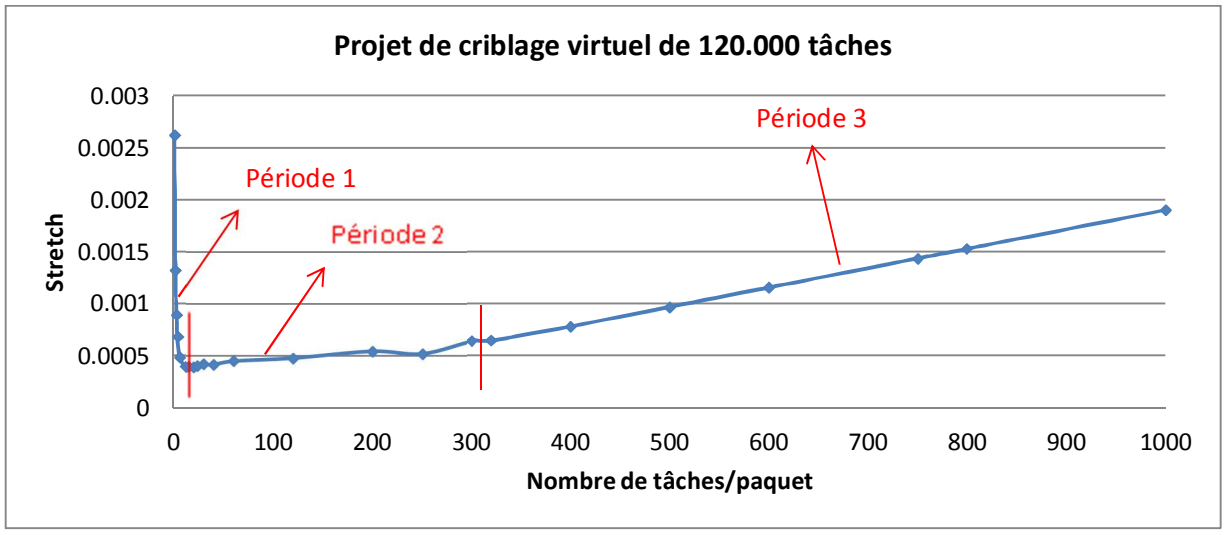


Figure 4-13: Résultat de simulation de l'impact du nombre de tâche/paquet sur le stretch

En accord avec les calculs de la section précédente, la simulation met en évidence trois zones marquées par des comportements différents en fonction du nombre de dockings dans un paquet :

- Zone ou période 1: Le stretch diminue inversement proportionnellement à x. Lorsque x est petit , ε (voir la formule 4.2) est petit.

- Zone ou période 2: Lorsque Ɛ n'est plus négligeable devant 1/x, le stretch augmente mais pas significativement.
- Période 3: Quand x > 300 tâches/ paquet, le stretch augmente linéairement avec x. On retrouve la formule 5.1 : N = 120.000 tâches et M = 400 machines ➔ N/M = 300.

Dans un deuxième test, nous considérons un projet plus petit de 1200 tâches à exécuter sur 400 machines. Le nombre de tâches par paquet prend les valeurs suivantes : 1, 2, 3, 4, 6, 12, 15, 20, 24, 30, 40, 60, 120.

On a $N_{j(i)}^{normal}$ = 1200 et M = 400 alors $N_{j(i)}^{normal}$ / M = 3

Le résultat obtenu est en très bon accord avec le modèle mathématique (Figure 4-14) :

- lorsque le nombre de tâches / paquet est supérieur à 3, le stretch augmente linéairement comme la formule 4.1.
- lorsque le nombre de tâches / paquet est inférieur à 3, le stretch diminue (formule 4.2)

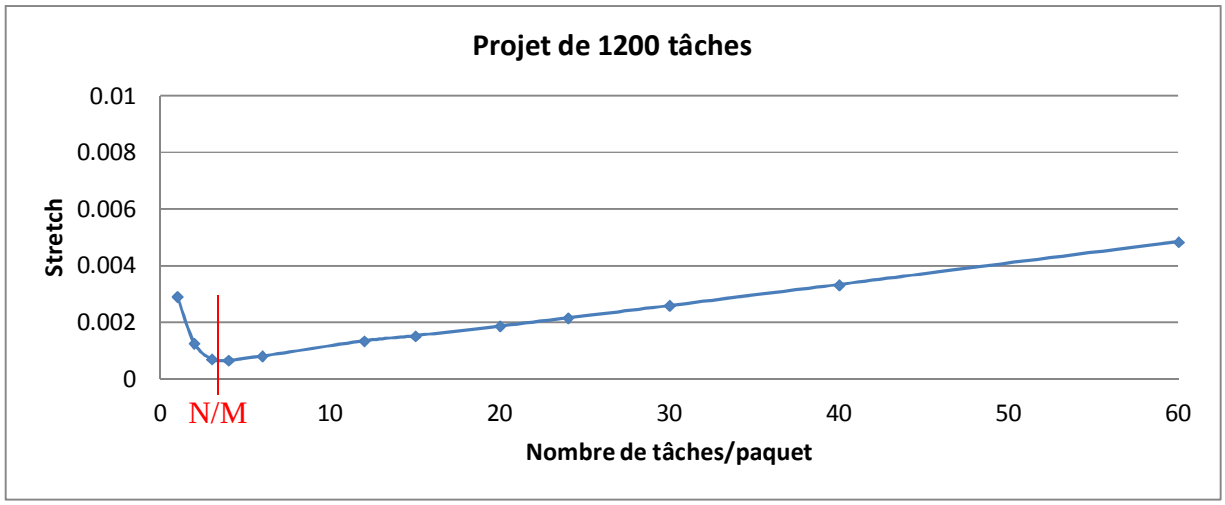


Figure 4-14: Résultat de simulation de l'impact du nombre de tâches par paquet sur le ralentissement

La Figure 4-15 montre l'impact du changement de temps de latence sur la variation du ralentissement en fonction du nombre de dockings dans un paquet pour le projet de 120.000 tâches. Les courbes correspondent à des temps de latence entre l'agent pilote et le gestionnaire de tâche de 8 secondes et 18 secondes.

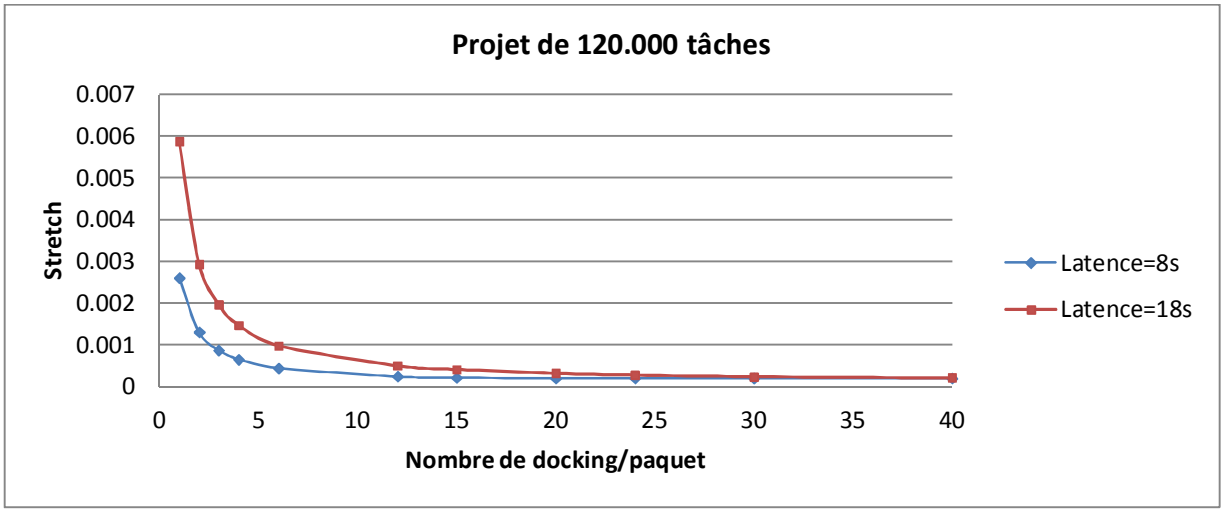


Figure 4-15: L'impact du nombre de tâches par paquet sur le ralentissement pour différents temps de latence

Le résultat obtenu reproduit le comportement attendu d'après la formule 4.3. Lorsque la latence est élevée, le ralentissement expérimenté par l'utilisateur dans la période 1 est plus élevé.

4.5 Conclusion du chapitre

Dans ce chapitre, nous avons décrit en détails l'environnement et les propriétés des tâches de criblage virtuel à déployer à travers les plates-formes sur la grille et le cloud. Sur la base de cette analyse, nous avons proposé une définition du problème étudié et introduit des notations qui seront utilisées dans les prochains chapitres.

Nous avons ensuite présenté l'environnement de simulation SimGrid avec lequel nous avons développé des modèles de plates-formes sur grille et cloud. Nous avons détaillé les modules utilisés et présenté une validation des comportements observés sur la grille lorsque les tâches de docking sont regroupées par paquet.

Nous allons présenter dans le prochain chapitre les performances des politiques d'ordonnancement obtenues avec ces modèles. Nous confronterons ensuite ces résultats à des expérimentations menées sur cluster et sur la grille de production.

CHAPITRE 5 : PERFORMANCES DES POLITIQUES D'ORDONNANCEMENT MODELISEES AVEC SIMGRID

Le fil directeur de ce manuscrit est le déploiement de calculs de criblage virtuel sur une infrastructure de grille ou de cloud depuis le Vietnam. Nous avons identifié la plate-forme à agents pilotes comme un outil privilégié pour ce déploiement, notamment parce qu'il permet de masquer aux utilisateurs, d'une part la complexité de la grille, d'autre part l'évolution technologique de la grille au cloud. Les ressources humaines de l'INPC ne lui permettent pas d'envisager de déployer et de maintenir son propre serveur, aussi nous sommes-nous intéressés aux problématiques d'ordonnancement des tâches soumises par une plate-forme à agents pilotes partagée par des communautés.

Dans ce chapitre, nous présentons des résultats obtenus avec SIMGRID sur des environnements simulés de grille et de cloud. Nous montrerons tout d'abord comment les paramètres du modèle sont tirés de données expérimentales collectées sur les environnements de production. Nous présenterons ensuite les résultats pour chaque environnement et tirerons des premières conclusions dont la validation expérimentale sera discutée dans le prochain chapitre.

5.1 Définition des paramètres du modèle de grille

Une des clefs de la pertinence du modèle est le choix de paramètres réalistes. Nous avons utilisé des données collectées sur des infrastructures de grille de production pour la latence de téléchargement des données par les agents pilotes des tâches soumises à la plate-forme, la

charge de la grille et les caractéristiques des projets soumis par les utilisateurs à la plate-forme à agents pilotes.

5.1.1 Latence de téléchargement des données

Nous avons vu au chapitre précédent comment la latence impactait le ralentissement expérimenté par les utilisateurs. Pour évaluer la latence de la connexion réelle, nous avons soumis à la grille à travers une plate-forme DIRAC EGI 10000 tâches simples dans lesquelles l'agent pilote télécharge un fichier vide à partir de l'élément de stockage. La figure 5-1 présente la distribution du temps de latence observé pour ces 10.000 tâches.

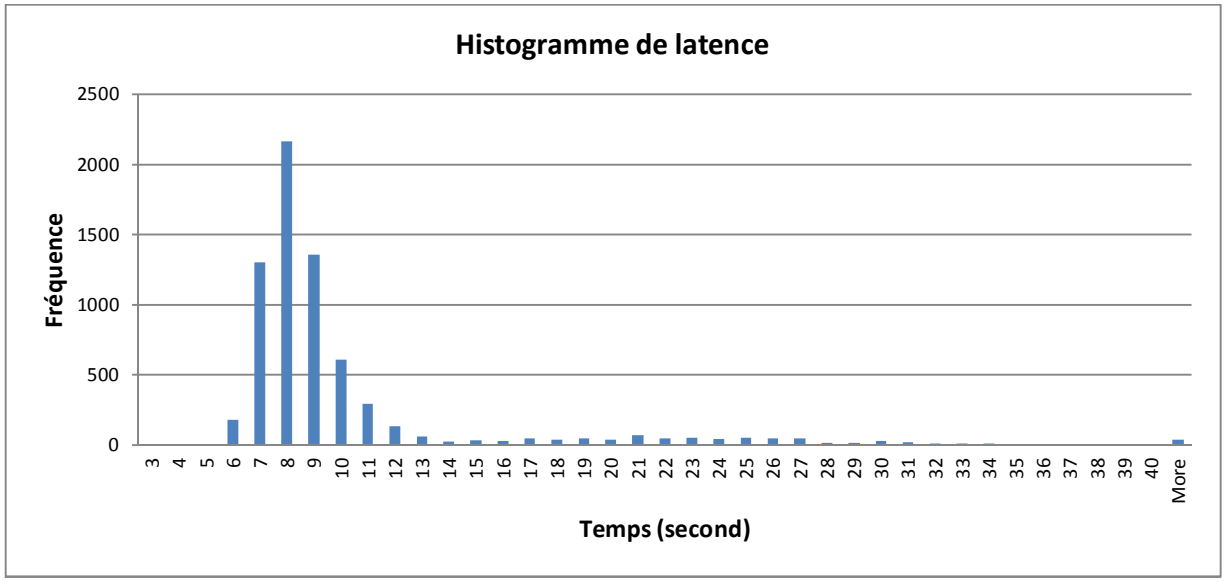


Figure 5-1 : Temps de latence en secondes observés pour 10.000 tâches soumises avec DIRAC

La latence la plus courte mesurée est 3.84s, la plus longue est 309.28s. 90% des latences mesurées se situent entre 3s et 16s avec une moyenne à environ 18s.

5.1.2 Charge de la grille

Pour simuler de façon réaliste la charge de la grille utilisée par la plate-forme à agents pilotes, nous avons utilisé les archives de l'infrastructure AuverGrid, grille régionale multidisciplinaire de la région Auvergne (France), dont la charge a été archivée pour la période 2004-2005 sur le site web de Grille Workload Archive [73]. Les éléments constituant l'infrastructure de la grille AuverGrid dans cette période sont détaillés dans le tableau 5-1, y

compris la vitesse relative des machines et la durée maximale autorisée d'un job exécuté sur un cluster. Toutes les machines d'un même cluster ont la même vitesse.

Nom du cluster	Nombre de nœuds de calcul (Worker Nodes)	Vitesse relative des nœuds de calcul	Durée maximale d'un job (secondes)
CLRLCGCE01	112	1	258220
CLRLCGCE02	84	1.1	258220
CLRLCGCE03	186	1.6	258220
IUT15	38	0.8	172800
OPGC	55	1.4	172800

Table 5-1 : La configuration de l'infrastructure de la grille AuverGrid

La figure 5.2 présente la charge de la grille Auvergrid pour les différentes périodes de 15 jours documentées dans les archives.

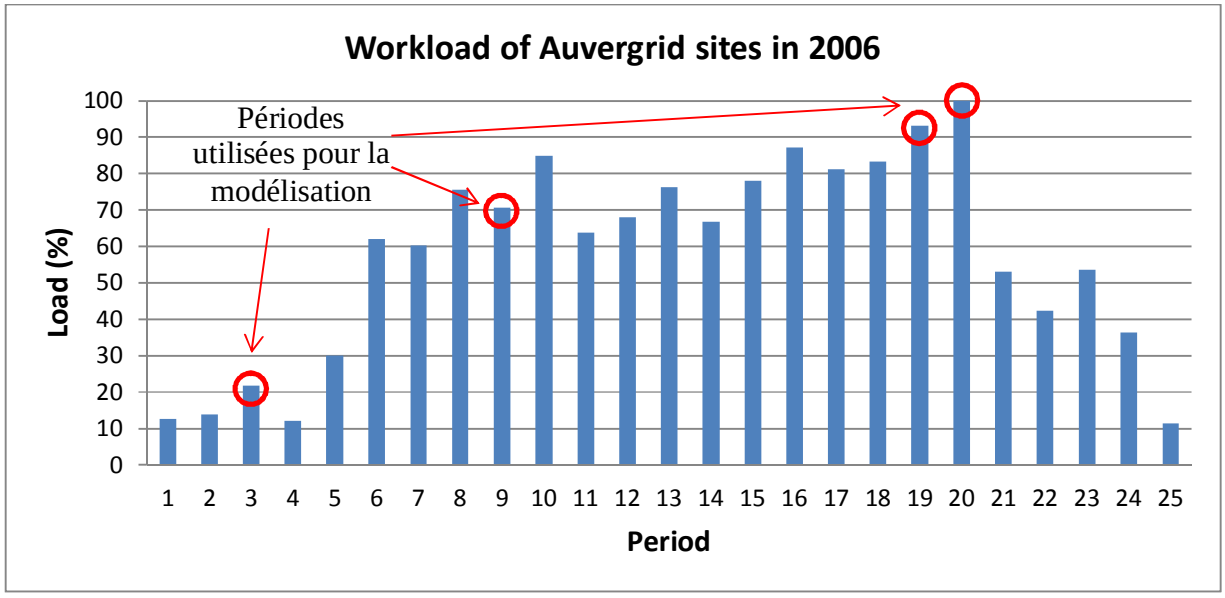


Figure 5-2: Archive de la charge des sites de la grille régionale Auvergrid

Nous avons utilisé 4 périodes caractérisées par des charges différentes pour évaluer l'impact de cette charge sur le ralentissement expérimenté par les utilisateurs : la période 3 avec une charge moyenne de 21,76%, la période 9 avec une charge moyenne de 70,67%, la période 19 avec une charge moyenne de 93,10%, la période 20 avec une charge de 100%.

5.1.3 Caractéristiques des projets soumis par les utilisateurs

L'analyse des charges de travail réelles de la grille dans [71], [74] a mis en évidence l'existence de deux catégories d'utilisateurs de la grille: les utilisateurs dits normaux et les gros utilisateurs dénommés « Data Challenge users ». Les utilisateurs normaux soumettent en général un nombre limité de jobs pour atteindre leurs objectifs scientifiques, alors que les utilisateurs « Data Challenge » soumettent de très grands nombres de jobs. Le nombre d'utilisateurs « Data Challenge » est limité, mais ils sont des clients importants de la plate-forme comme ils travaillent souvent pour le compte d'une collaboration scientifique.

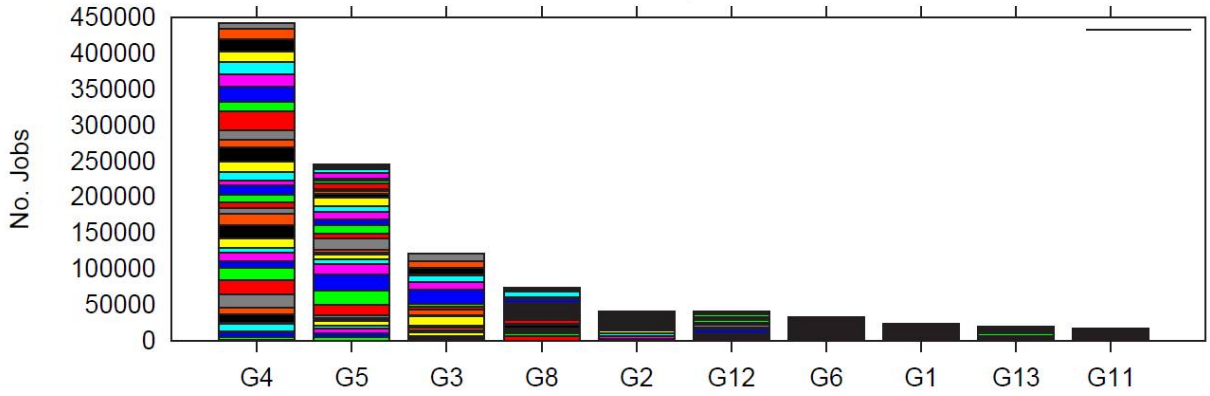


Figure 5-3: Nombre de jobs soumis par les 10 groupes d'utilisateurs les plus actifs d'un cluster de la grille du LHC de Mai 2005 à Janvier 2006. Source: An Analysis of Four Long-Term Grid Traces- [71]

Ces deux catégories d'utilisateurs sont observées dans la communauté de physique des particules. A titre d'exemple, la figure 5-3 représente le nombre de jobs soumis au cours d'une période de 6 mois de mai 2005 à janvier 2006 par les 10 groupes d'utilisateurs les plus actifs de la grille du LHC [71]. Trois groupes, G4, G5 et G3, soumettent la plupart de jobs sur ce cluster tandis que les autres groupes soumettent un nombre de jobs beaucoup plus petit.

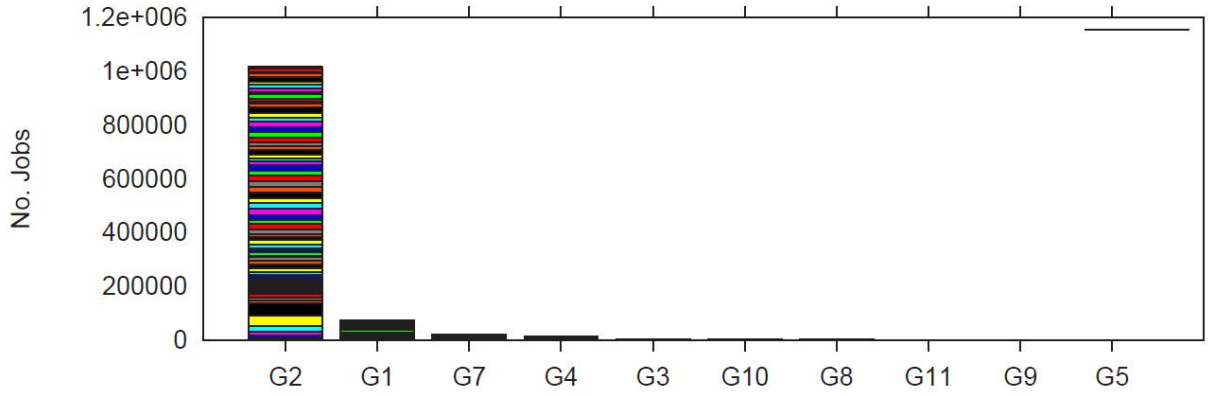


Figure 5-4: Nombre de jobs soumis par les 10 groupes d'utilisateurs les plus actifs d'un cluster de TeraGrid. Source : An Analysis of Four Long-Term Grid Traces - [71]

L'analyse de la charge sur un cluster de la grille TeraGrid, représentée sur dans la figure 5-4, met en évidence le même comportement. Un groupe (G2) soumet la plupart des jobs tandis que les autres groupes ne soumettent qu'un petit nombre de jobs.

Plus récemment, la figure 5-5 montre le nombre de jobs soumis par les plus gros utilisateurs sur le serveur FG-DIRAC de Juin 2012 à Septembre 2014. Quatre utilisateurs représentent 70% des jobs soumis.

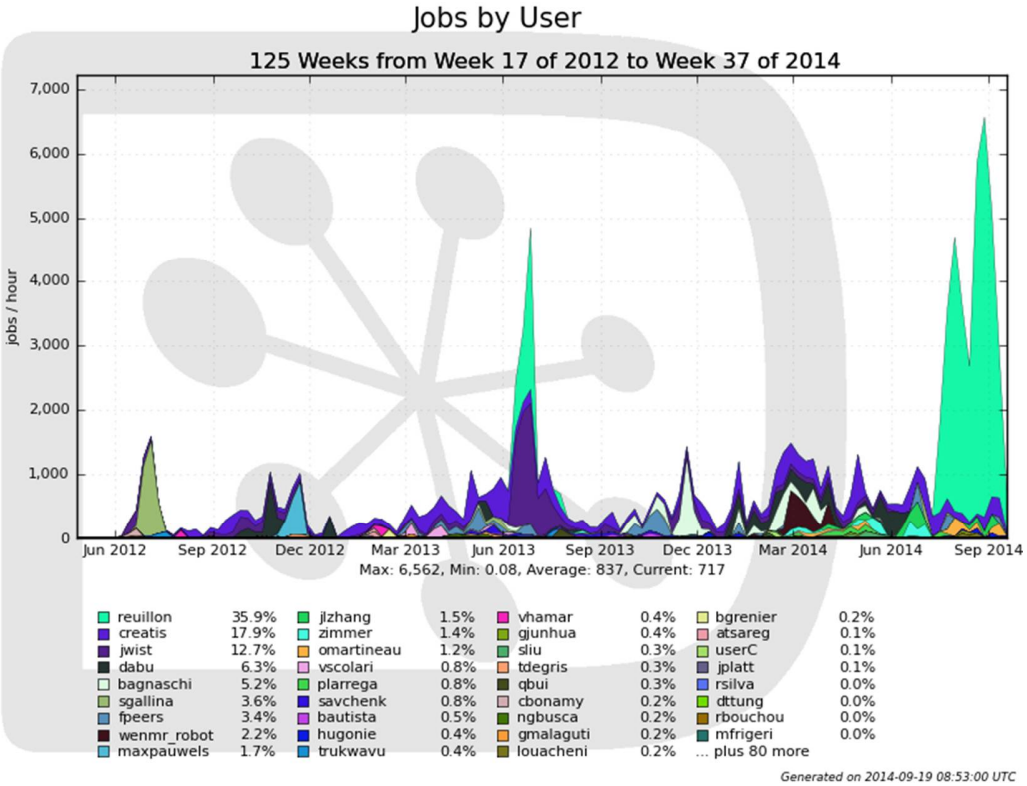


Figure 5-5: Nombre de jobs soumis par heure par les plus gros utilisateurs du serveur FG-DIRAC de Juin 2012 à Septembre 2014 (source : http://dirac.france-grilles.fr/DIRAC/)

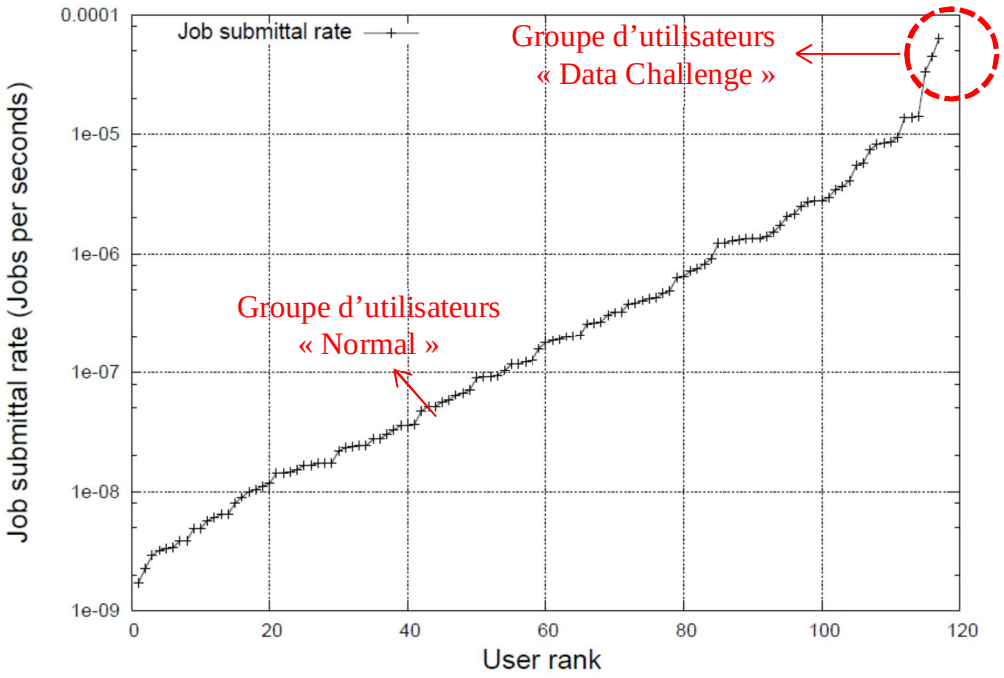


Figure 5-6: Analyses de la charge de la VO Biomed (Source: Workload analysis of a cluster in a grid environment)

Le même comportement est observé sur la figure 5-6 qui représente la charge de l'organisation virtuelle Biomed d'EGI exprimée en taux de soumission de jobs par utilisateur. Cette analyse met en évidence deux comportements : les utilisateurs normaux et quelques utilisateurs dont le taux de soumission des jobs est beaucoup plus élevé.

Au vu de ces comportements, en nous inspirant de la littérature, nous avons choisi de modéliser la distribution des tâches soumises à la plate-forme en considérant deux groupes d'utilisateurs, les utilisateurs normaux et les utilisateurs « data challenge ». Chaque groupe est composé de plusieurs types d'utilisateurs caractérisés par un intervalle de temps moyen λ d'arrivée entre deux utilisateurs et un nombre moyen γ de jobs soumis. Pour reproduire le comportement observé sur la figure 5-6, nous considérons que, dans chaque groupe, l'intervalle de temps moyen entre deux utilisateurs de type i est égal à $\lambda = c \times d^i$ et que chaque utilisateur de type i soumet un nombre de tâches égal à $\gamma = a \times b^i$. Les valeurs des paramètres a, b, c et d sont ajustés pour les deux groupes normaux et « data challenge » sur les observations expérimentales de la figure 5-6.

L'intervalle entre deux utilisateurs consécutifs de type i est généré suivant une distribution de Poisson [75], [76] de moyenne λ :

$$f(x) = \frac{1}{\lambda} e^{-x.\frac{1}{\lambda}} \text{ avec x >= 0}$$

Les temps d'arrivée des utilisateurs sont ainsi générés jusqu'à une limite fixée à 400.000 secondes, soit 4,6 jours.

De façon similaire, le nombre de tâches de docking soumis par un utilisateur de type i est généré selon une loi de distribution géométrique [74], [77] de raison p = 1/γ, La probabilité qu'un utilisateur soumette un nombre k de jobs est égale à : $Pr(X=k) = (1-p)^{k-1}p$.

De cette façon, on peut générer de façon aléatoire des exemples de charge de la grille qui suivent le comportement expérimental observé (figure 5-6) en ajustant les paramètres a, b, c et d.

En conséquence, pour le groupe « Normal », nous utilisons $a = 1, b = \sqrt[40]{20}$ pour générer les tâches de docking dans chaque projet. Et nous supposons qu'un nouveau projet arrive à la plate-forme au moins après 10 minutes (600 secondes), de telle sorte que : $c = 600, d = \sqrt[40]{20}$.

Pour le groupe « Data Challenge », les utilisateurs préparent des projets beaucoup plus gourmands en ressource. Nous utilisons $a = 60000, b = \sqrt[20]{10}$ pour générer le nombre de tâches de docking soumises par chaque utilisateur. Nous supposons qu'un nouveau utilisateur arrive à la plate-forme au moins après 30000 secondes (8h20'), donc nous avons choisi : $c = 30000, d = \sqrt[20]{10}$.

Pour simuler différentes configurations, nous avons considéré 4 scénarii correspondant à des populations différentes des groupes normaux et « data challenge » (table 5.2). Pour chaque scénario ou cas, nous avons généré 500 exemples de charge de la grille grâce aux tirages aléatoires des temps d'arrivée et des nombres de tâches soumises par les utilisateurs dans les distributions aléatoires décrites précédemment.

	Nombre de type d'utilisateurs du groupe « Normal »	Nombre de type d'utilisateurs du groupe « Data Challenge »
Cas 00	199	1
Cas 01	195	5
Cas 02	190	10
Cas 03	185	15

Table 5-2: Nombre d'utilisateurs normaux et « data challenge » dans les 4 scénarios considérés

Nous disposons ainsi de quatre jeux de données de 500 exemples de la charge des utilisateurs de criblage virtuel, dont les caractéristiques sont résumées dans le tableau 5-3. Du cas 00 au cas 03, la proportion du nombre total de tâches soumises par les « data challenge » utilisateurs passe de 65% à 99%.

Dataset	Nombre d'utilisateur dans le groupe « normal »	Nombre d'utilisateur dans le groupe "Data Challenge"	Nombre total de tâches soumises par le groupe "normal user"	Nombre total de tâches soumises par le groupe « Data Challenge user »
Cas 00	9.352	5	176.671	327,355
Cas 01	9.347	26	146.496	1.976.917
Cas 02	9.341	54	121.861	5.273.020
Cas 03	9.335	86	103.213	10.841.076

Table 5-3: Résumé des 4 jeux de données

Le dernier paramètre à définir est la durée des tâches de docking. Nous avons vu au chapitre précédent que, pour une cible donnée, cette durée dépendait de la taille du ligand. Nous avons cependant souhaité nous ramener à un problème d'ordonnancement dans lequel la durée des tâches était constante.

La simulation nous permettant de considérer plus facilement plusieurs cas de figures, nous avons étudié les deux options : soit des tâches de docking de durée constante, soit des tâches de durée variable. Dans ce dernier cas, nous avons utilisé la liste des 1392 tâches de docking soumises pour des ligands de la base de données ZINC d'intérêt pour la maladie d'Alzheimer, parce que la variation du temps de calcul est plus grande dans le cas du projet sur la maladie d'Alzheimer (27.85%) que dans le cas du projet sur la Dengue (20.44%). Les temps d'exécution pour les 1392 tâches ont été enregistrés dans une table indexée. Ensuite, nous utilisons une variable aléatoire X qui reçoit des valeurs entières entre 1 et 1392 selon une distribution uniforme. A partir de la valeur de X, nous déduisons de la table le temps de calcul correspondant de la tâche de docking dans la simulation. Pour chaque utilisateur, nous avons effectué autant de tirages aléatoires que de tâches soumises par cet utilisateur.

Pour les tâches de durée constante, nous avons opté pour une durée de 484.44 secondes, dans la moyenne des temps d'exécution observés dans les trois projets étudiés.

5.2 Evaluation des politiques d'ordonnancement sur la grille

Nous considérons le cas 00 décrit ci-dessus dont nous simulons 500 exemples en changeant la politique d'ordonnancement dans le module Task Manager de la plate-forme d'agent pilote. Nous considérons 5 politiques d'ordonnancement qui ont été présentées dans le chapitre 2 :

- First In First Out (FIFO), utilisé par la plate-forme WISDOM. L'ordonnancement est fait dans l'ordre d'arrivée en gérant une file unique des processus sans priorité ni préemption. Utilisé par la plate-forme WISDOM.
- Round Robin (RR), utilisé par la plate-forme DIRAC. L'ordonnanceur sélectionne tour à tour un job d'un utilisateur différent dans la file d'attente.
- Longest Processing Time (LPT). Les jobs sont triés par ordre de durée décroissante : les jobs les plus longs sont exécutés en premier. Cette politique n'est pas déployée sur la grille.
- Shortest Processing Time (SPT). Les jobs sont triés par ordre de durée croissante : les jobs les plus courts sont exécutés en premier. Cette politique n'est pas déployée sur la grille

Pour chaque politique (FIFO, SPT, LPT, RR et FS), nous calculons sur les 500 exemples les ralentissements moyens et maximaux Smoyen et Smax de tous les utilisateurs afin de comparer ces politiques.

Pour rappel, si N est le nombre d'utilisateurs, on calcule le ralentissement S de chaque utilisateur i par la formule :

$$S_i = \frac{Temps\ de\ calcul\ du\ projet\ sur\ la\ plateforme}{Temps\ de\ calcul\ sur\ 1\ CPU} \quad (i = 1..N)$$

Le ralentissement moyen S_{moyen} est défini comme la moyenne des ralentissements expérimentés par les utilisateurs: $S_{moyen} = \frac{\sum_{i=1..N} (S_i)}{N}$

Le ralentissement maximal S_{max} est le ralentissement maximal expérimenté par les utilisateurs :

$S_{max} = max_{i=1..N} (S_i)$

Du fait du choix de ces définitions, les ralentissements peuvent être inférieurs à 1.

Les figures 5-7 et 5-8 représentent les ralentissements observés pour les 5 politiques dans le cas de tâches de durées variables (figure 5-7) et de durée constante (figure 5-8).

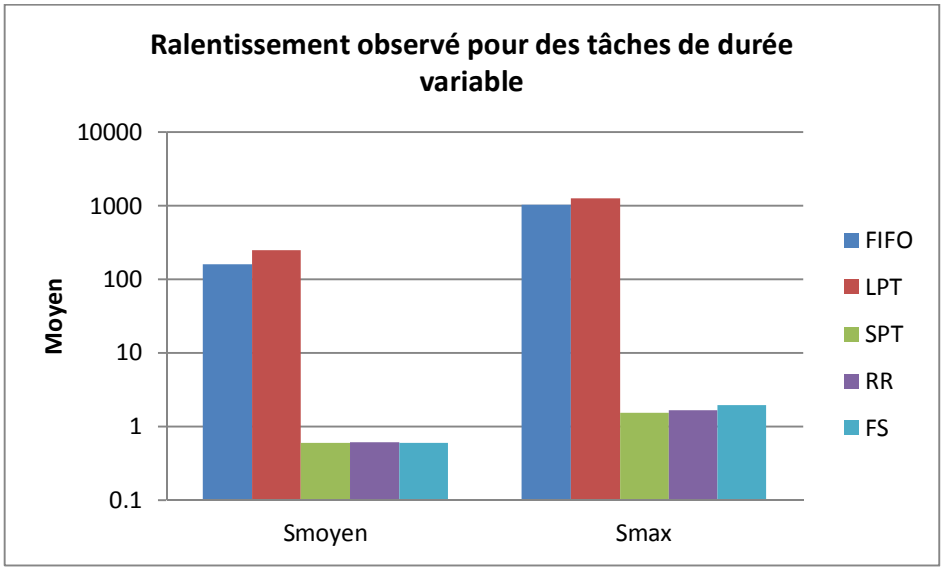


Figure 5-7: Ralentissements S_{moyen} et S_{max} observés pour des tâches de durée variable avec 5 politiques sur la grille

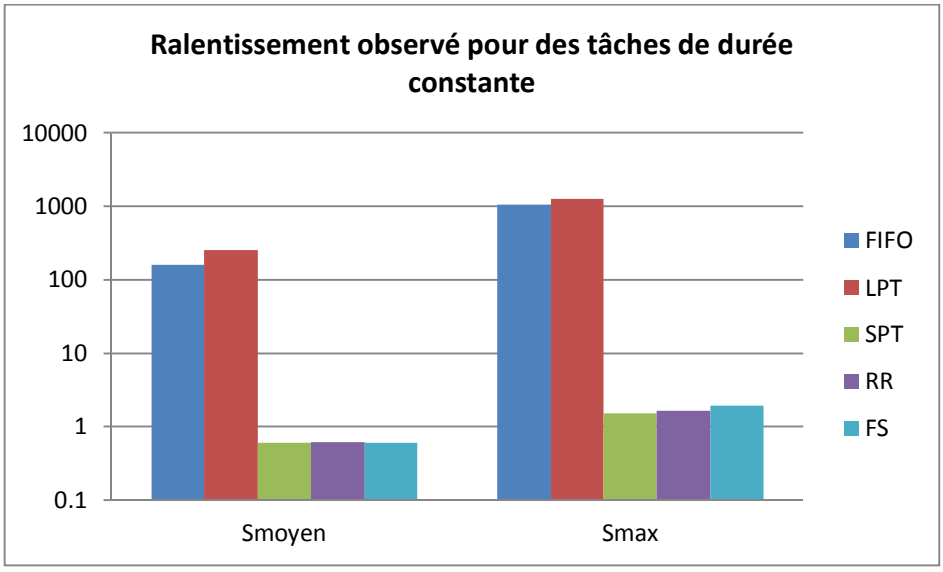


Figure 5-8: Ralentissements S_{moyen} et S_{max} observés pour des tâches de durée constante avec 5 politiques sur la grille

SPT est une très bonne politique d'ordonnancement qui obtient les meilleurs résultats aussi bien pour le ralentissement moyen que pour le ralentissement maximal. Les politiques Fair Share et Round Robin obtiennent des résultats satisfaisants tandis que FIFO et LPT ont des résultats catastrophiques par rapport aux autres politiques. Ces résultats étaient attendus : FIFO est ainsi un mauvais choix par rapport à une politique d'ordonnancement de l'équité parce que les utilisateurs avec un nombre limité de tâches doivent attendre plus longtemps s'ils arrivent juste après les utilisateurs avec de nombreuses tâches. De même LPT, en privilégiant les utilisateurs dont les jobs sont plus longs, va pénaliser fortement les utilisateurs dont les jobs sont courts.

La comparaison des figures 5-7 et 5-8 met en évidence une similarité des résultats pour des tâches constantes et des tâches de durée variable.

Les tableaux 5-4 et 5-5 proposent une comparaison deux à deux des politiques d'ordonnancement pour chacun des 500 exemples générés avec des tâches de durée variable. Dans chaque case figure le nombre de fois sur 500 où la politique en colonne a été plus performante que la politique de la ligne. Par exemple, en ce qui concerne le ralentissement moyen, LPT est systématiquement la plus mauvaise politique tandis qu'en termes de ralentissement maximal, elle est meilleure que FIFO dans 57 exemples.

	FIFO	**SPT**	**LPT**	**RR**	**FS**
FIFO		500	0	500	500
SPT	0		0	7	222
LPT	500	500		500	500
RR	0	493	0		478
FS	0	278	0	12	

Table 5-4: Comparaison de S_{moyen} entre les 5 politiques

	FIFO	**SPT**	**LPT**	**RR**	**FS**
FIFO		500	57	500	500
SPT	0		0	106	78
LPT	443	500		500	500
RR	0	394	0		144
FS	0	422	0	356	

Table 5-5: Comparaison de S_{max} entre les 5 politiques

En ce qui concerne le ralentissement moyen, la politique SPT est meilleure que FIFO et LPT dans tous les 500 cas, meilleure que RR dans la plupart des cas (493/500) et meilleure que FS dans la moitié des cas (278/500). La politique FS est meilleure que RR dans la plupart des cas (478/500).

En ce qui concerne le ralentissement maximal, la politique SPT est meilleure que FIFO et LPT dans tous les 500 cas. SPT est meilleure que RR et FS dans la plupart des cas (394/500 pour RR et 422/500 pour FS). RR est meilleure que FS dans plus de la moitié des cas (356/500).

5.2.1 Impact de la charge de la grille

Nous avons étudié la variation des ralentissements [S_{max}, S_{moyen}] générés par les politiques d'ordonnancement en fonction de la charge de la grille en considérant 4 périodes différentes de 15 jours (périodes 3, 9 , 19 et 20 de la figure 5-2) où les charges sont estimées respectivement à 21,76%, 70,67%, 93,10% et 100%.

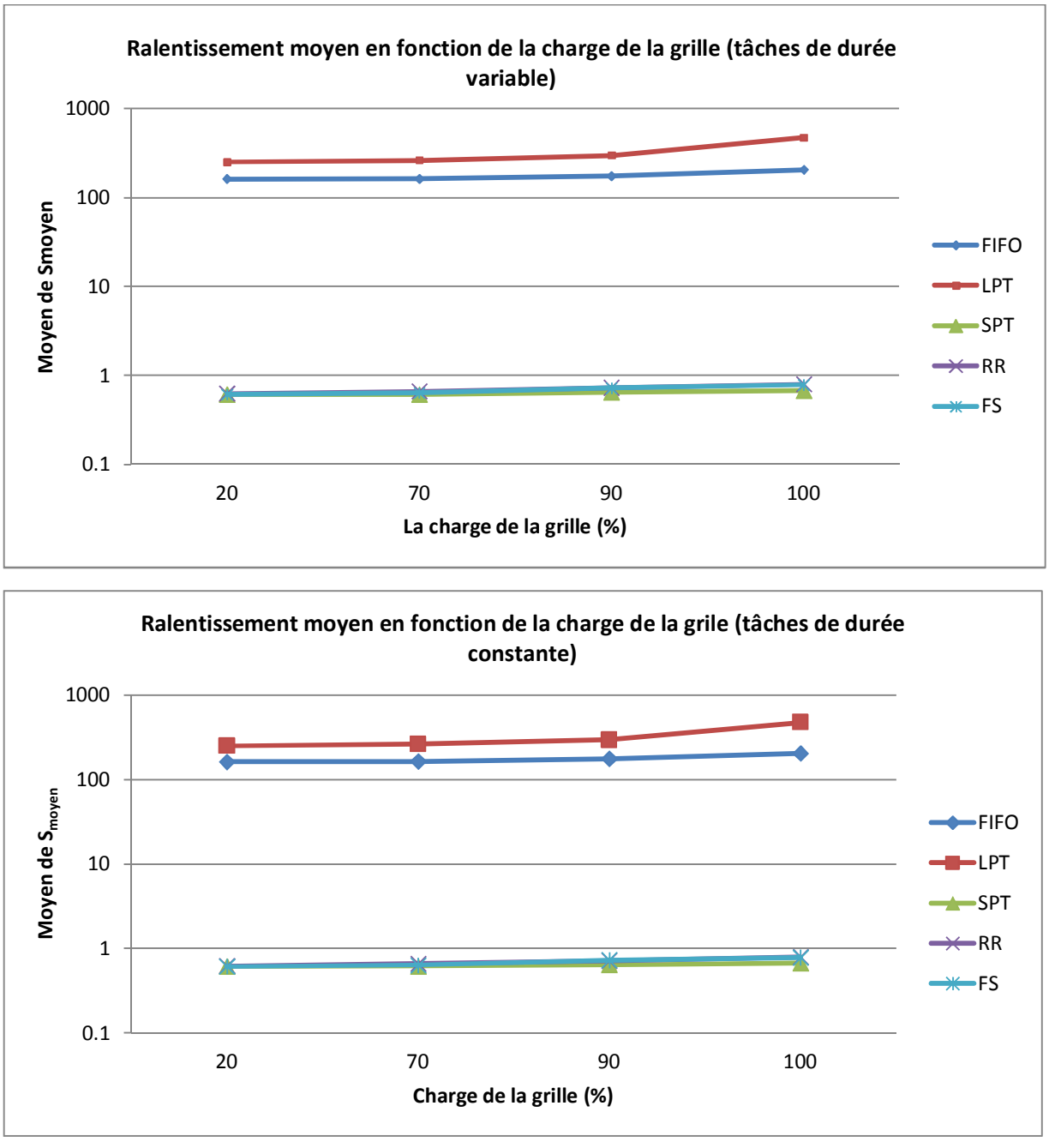


Figure 5-9: Ralentissement moyen en fonction de la charge de la grille pour des tâches de durée variable et constante

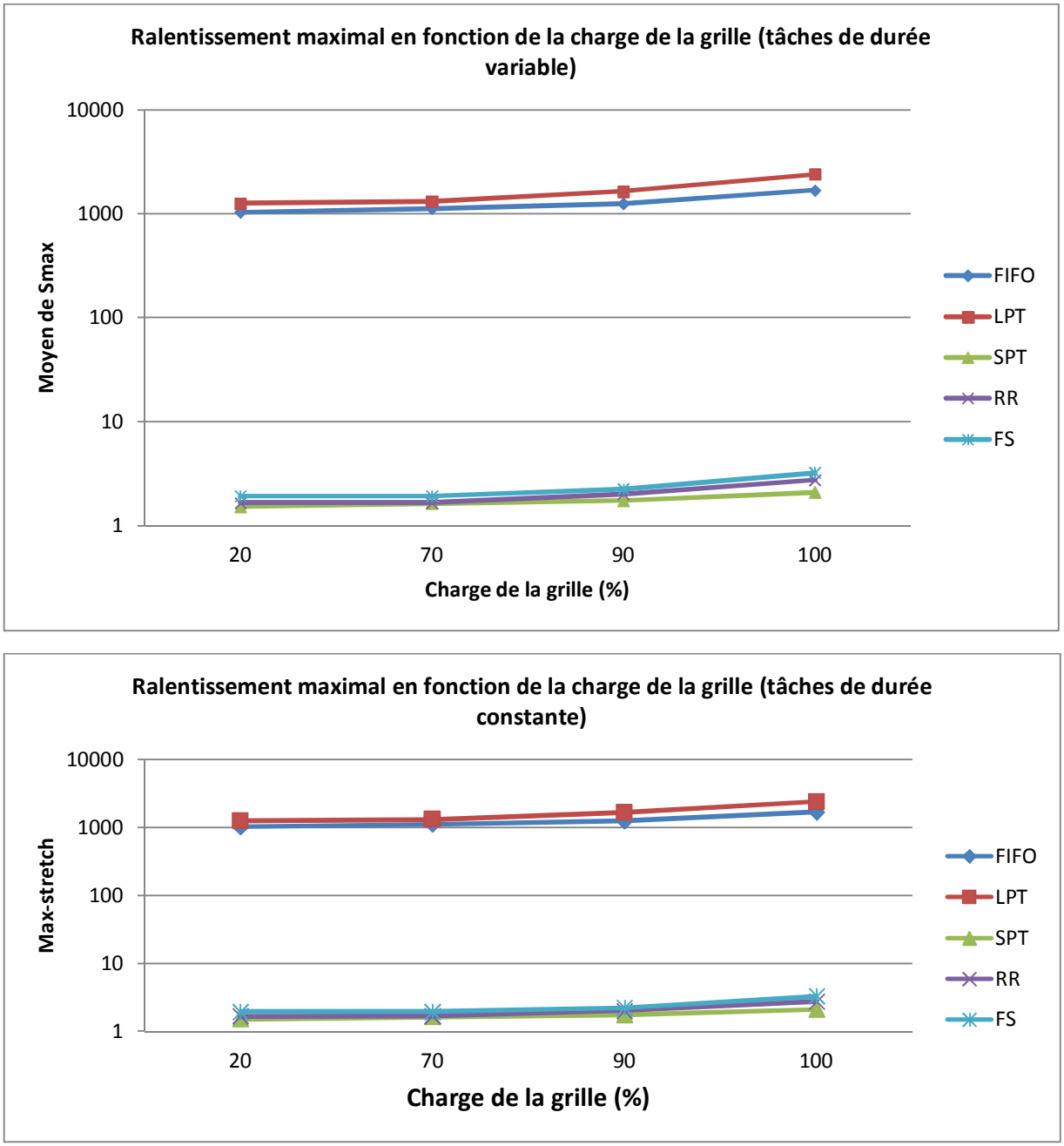


Figure 5-10: Ralentissement maximal en fonction de la charge de la grille pour des tâches de durée variable et constante

Les figures 5-9 et 5-10 montrent les résultats de simulation pour S_{moyen} et S_{max} lors du changement de la charge de l'environnement de la grille. A partir des temps d'arrivée et des temps d'achèvement des tâches de docking dans la simulation, on peut évaluer la charge générée par les utilisateurs de criblage virtuel. En utilisant la formule au paragraphe 4.3.2.2, elle représente 8.95% de la puissance de calcul de la grille. C'est pourquoi l'augmentation de la charge de la grille de 21,76% à 70,67% ne change pas significativement le ralentissement

expérimenté par les utilisateurs. Toutefois, lorsque la charge de la grille atteint le niveau de 93,1% à 100%, le ralentissement augmente significativement. Les tables 5-6 et 5-7 montrent l'évolution du ralentissement des utilisateurs.

Max du stretch					
Charge de la grille (%)	FIFO	LPT	SPT	RR	FS
21,76	1033,71	1265,37	1,53	1,67	1,93
70,67	1123,25	1312,57	1,55	1,81	1,99
93,1	1253,81	1643,68	1,75	2,03	2,26
100	1687,56	2403,67	2,10	2,75	3,25

Table 5-6: Ralentissement maximal des utilisateurs en fonction de la charge de la grille

Moyen du stretch					
Charge de la grille (%)	FIFO	LPT	SPT	RR	FS
21,67	162,09	250,98	0,61	0,62	0,61
70,67	163,03	263,94	0,61	0,66	0,64
93,1	175,59	297,94	0,64	0,73	0,72
100	205,50	474,59	0,67	0,80	0,78

Table 5-7: Ralentissement moyen des utilisateurs en fonction de la charge de la grille

Dans tous les cas, SPT reste la meilleure politique d'ordonnancement entre les 5 politiques testés sur le critère du ralentissement et la moins sensible à la charge de la grille.

5.2.2 Impact de la durée maximale d'exécution sur les éléments de calcul

Afin de simuler l'impact de la disponibilité de la machine sur le ralentissement des utilisateurs, nous avons modifié le temps de calcul maximal des clusters. En effet, les agents pilotes soumis avec succès à des éléments de calcul de la grille ont une durée de vie limitée à ce temps de calcul maximal.

Pour modifier la durée maximale de calcul des clusters de l'infrastructure AuverGrid telle qu'elle est documentée dans le tableau 5-1, nous avons multiplié ce temps maximal par un facteur X prenant les valeurs 0,001, 0,01, 0,1 et 1. Les figures 5-11 et 5-12 montrent l'impact de la variation de ce paramètre X sur les performances des différentes politiques d'ordonnancement. Pour cette simulation, nous considérons que la charge de la grille est de 21.76 %.

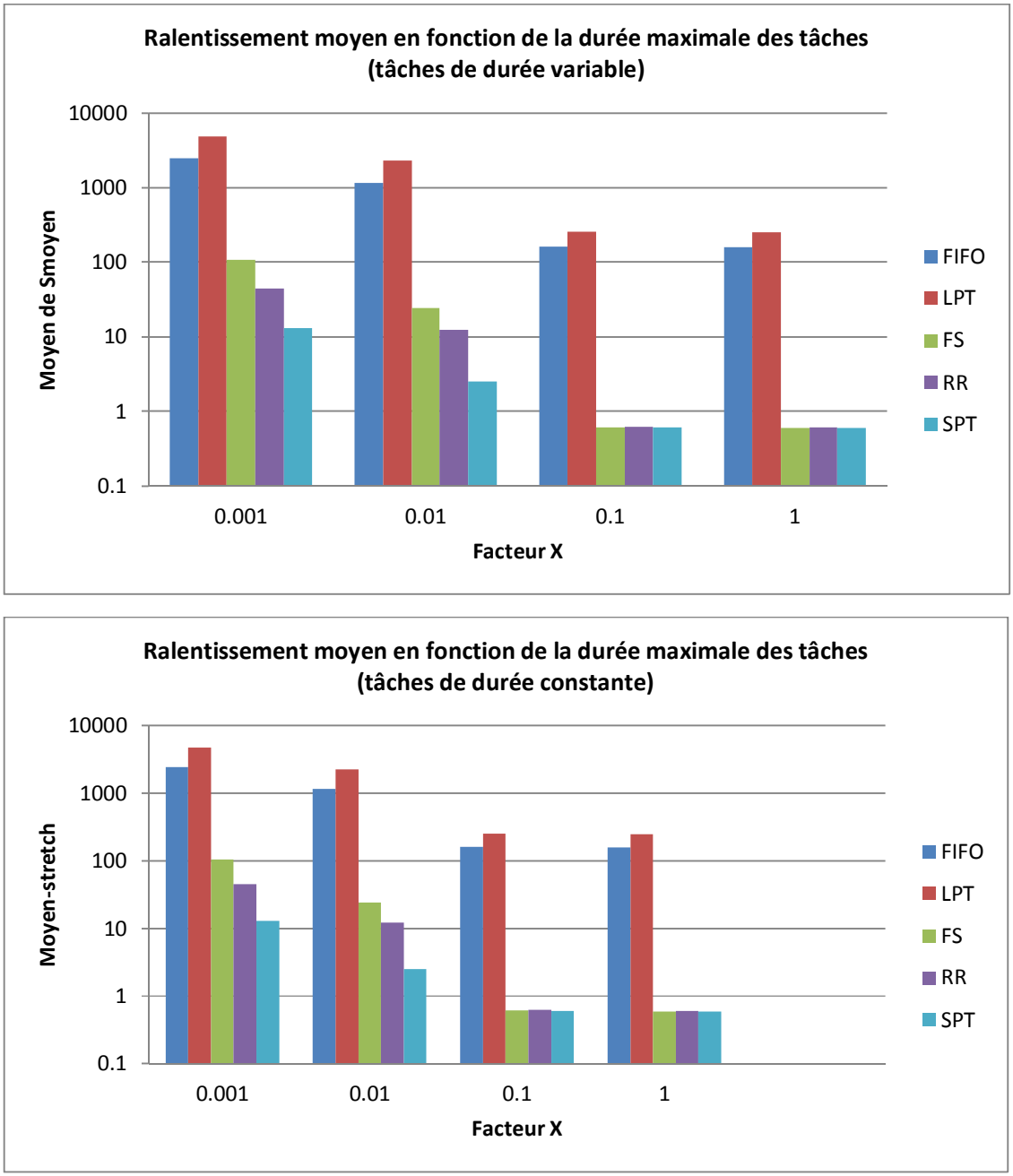
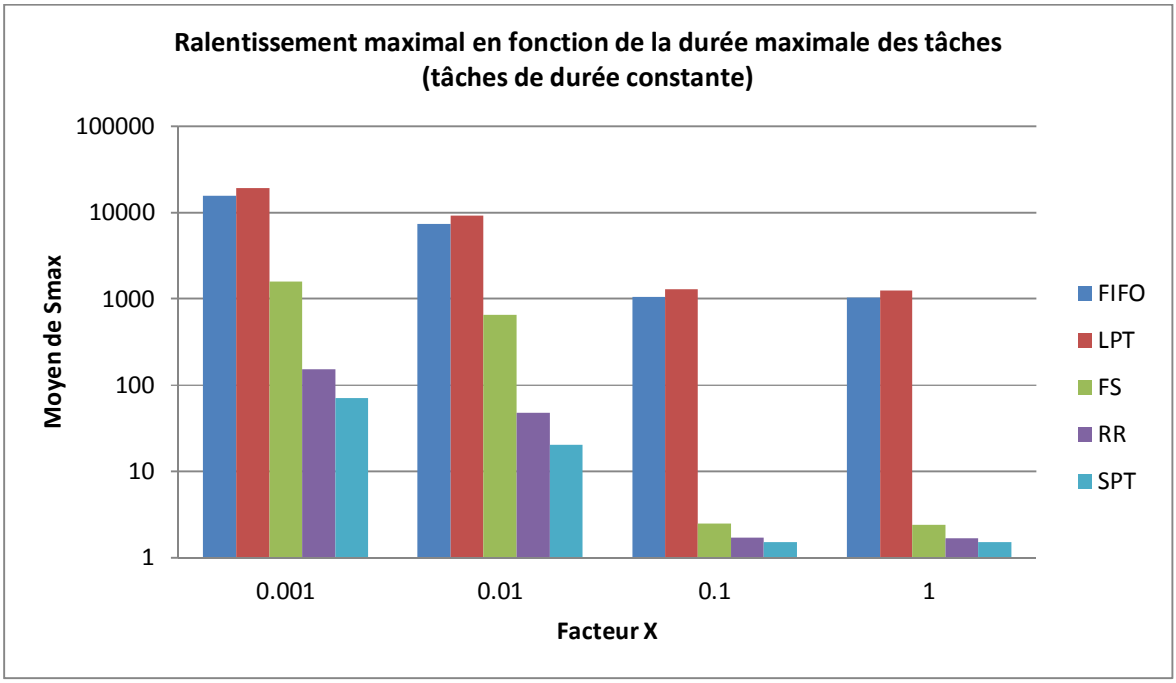


Figure 5-11:Impact sur le ralentissement moyen des politiques d'ordonnancement de la réduction du temps maximal de calcul sur les éléments de calcul de la grille



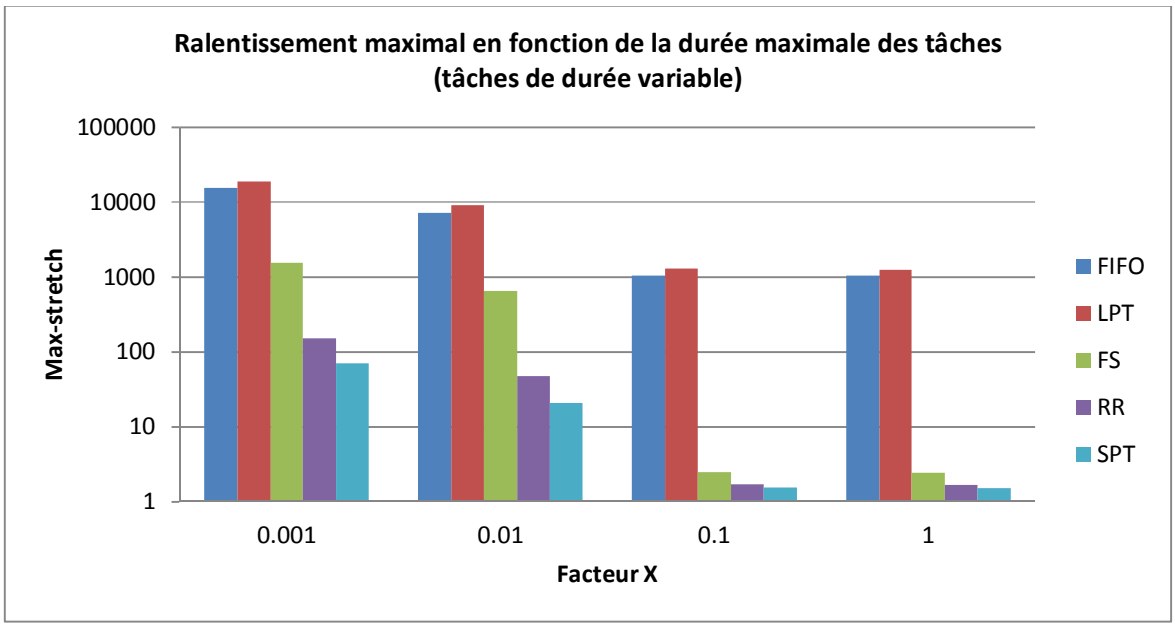


Figure 5-12:Impact sur le ralentissement maximal des politiques d'ordonnancement de la réduction du temps maximal de calcul sur les éléments de calcul de la grille

Nous observons comme attendu que, lorsque le temps de calcul maximal augmente (X varie de 0,001 à 1), le ralentissement des utilisateurs diminue pour toutes les politiques d'ordonnancement. En effet, lorsque le temps de calcul des agents de pilotes est court, plusieurs agents pilotes sont nécessaires pour exécuter toutes les tâches d'un utilisateur et le temps perdu augmente (ref. la section 4.3.2.3). Lorsque le temps de calcul maximal diminue, la politique SPT reste la plus performante.

En conclusion, quels que soient la configuration des sites et le niveau de charge de la grille, les résultats de simulation montrent que la politique SPT est toujours la meilleure parmi les 5 politiques pour optimiser le ralentissement expérimenté par les utilisateurs.

5.2.3 Impact de politique spécifique aux groupes d'utilisateurs (SPT-RR, SPT-SPT)

Nous avons démontré que SPT a les meilleurs résultats parmi les cinq politiques étudiées. Elle n'est pourtant implémentée par aucune des plates-formes actuelles. En effet, comme nous l'avons vu précédemment, la recherche sur la charge réelle de la grille [71], [74] a mis en évidence l'existence de deux types d'usagers de la grille: les utilisateurs normaux qui présentent souvent des petits nombres de jobs à la grille et les utilisateurs « data challenge » qui soumettent occasionnellement mais de grands nombres de jobs sur la grille. A ralentissement égal, un utilisateur soumettant 100.000 tâches va devoir attendre 1000 fois plus qu'un utilisateur soumettant 100 tâches. La politique SPT, en particulier, n'est pas appropriée pour les utilisateurs « Data Challenge » parce qu'ils doivent toujours attendre après les utilisateurs normaux.

Pour gérer cette difficulté, nous avons donc proposé une nouvelle politique qui utilise une technique d'ordonnancement différentiée des files d'attente des utilisateurs normaux et « data challenge » pour la planification dans la plate-forme d'agents pilotes. Dans cette nouvelle politique, l'administrateur crée deux groupes d'utilisateurs séparés dans la plate-forme: le groupe normal et le groupe « Data Challenge ». Chaque groupe a sa propre file d'attente de tâches sur la plate-forme. Un paramètre p ($p \in [0,1]$) est assigné à une file d'attente de tâches et (1-p) à l'autre. Ce paramètre est la probabilité que la file d'attente des tâches soit choisie pour envoyer ses tâches aux agents pilotes. La politique SPT est appliquée sur la file d'attente des tâches d'utilisateur normal. En file d'attente des tâches d'utilisateur « Data Challenge », nous testons des politiques SPT et RR pour sélectionner la plus adaptée. Nous appelons SPT-SPT la politique d'ordonnancement dans laquelle les deux groupes d'utilisateurs utilisent la politique SPT et SPT-RR la politique d'ordonnancement dans laquelle le groupe d'utilisateurs normaux utilise la politique SPT et le groupe « Data Challenge » utilise la politique RR.

Nous avons testé des politiques SPT-RR et SPT-SPT sur la configuration de la grille Auvergrid avec la charge de 21,76%.

Les figures 5-13 et 5-14 présentent une comparaison des performances de la politique SPT-RR pour différentes valeurs de p avec les politiques d'ordonnancement les plus performantes (FS, RR et SPT) pour les deux groupes d'utilisateurs (utilisateurs normaux et utilisateurs « Data Challenge ») pour des tâches de durée constante et variable. Les 4 scénarios ont été considérés (cas 00 à cas 03) avec des contributions croissantes des utilisateurs data challenge.

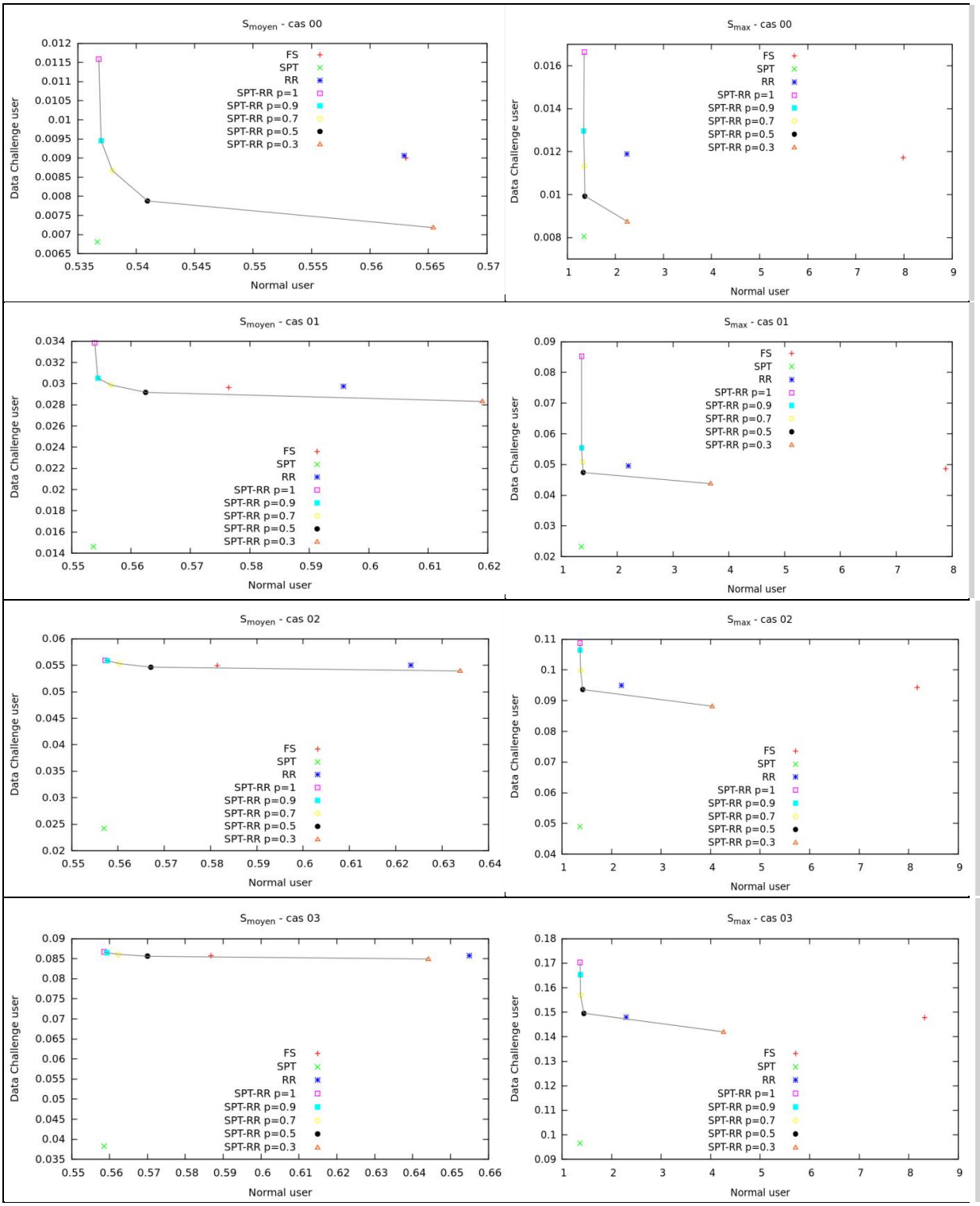


Figure 5-13: Comparaison des performances en termes de ralentissements moyen et maximal de la politique SPT-RR avec les autres politiques d'ordonnancement pour des tâches de durée variable

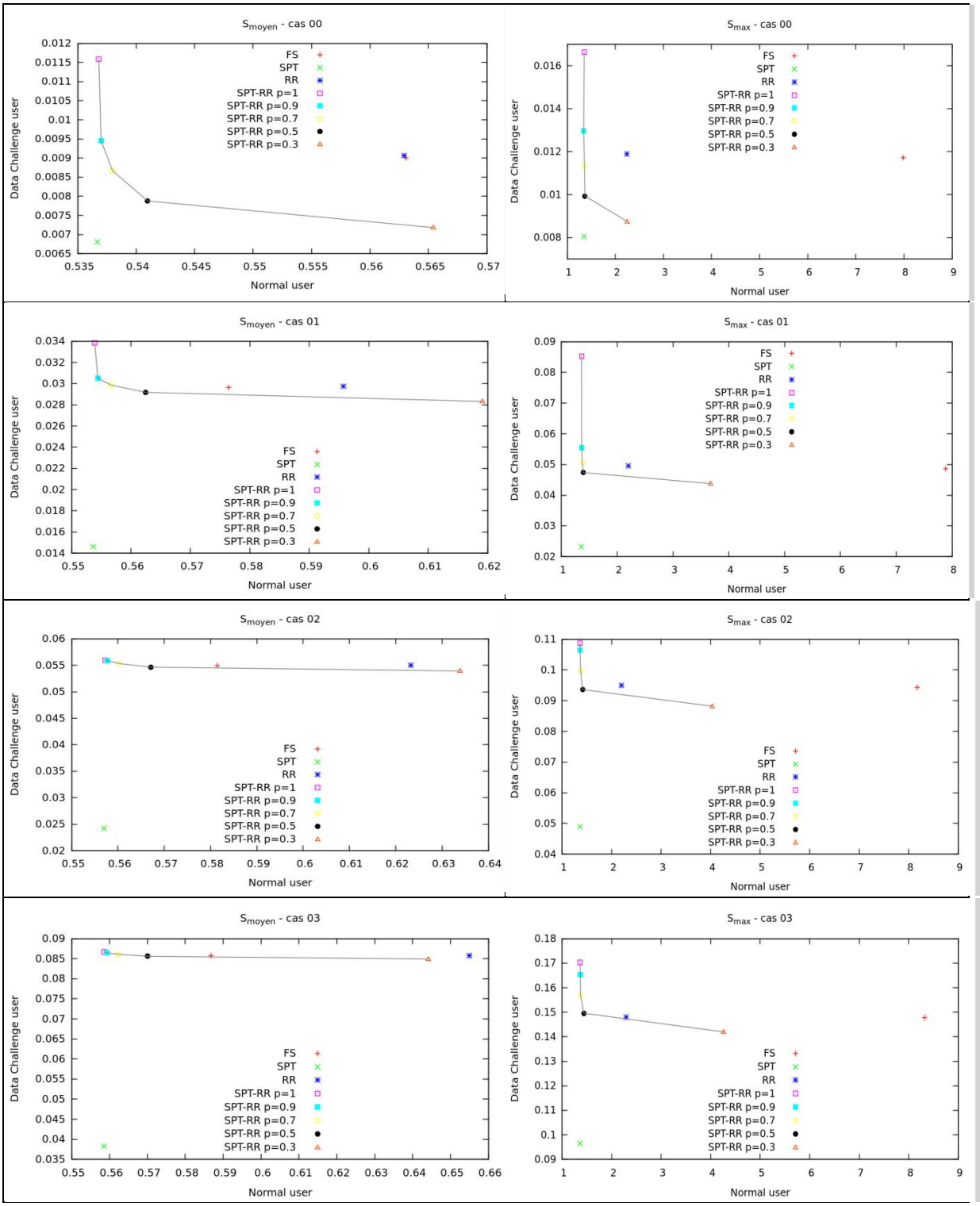


Figure 5-14: Comparaison des performances en termes de ralentissements moyen et maximal de la politique SPT-RR avec les autres politiques d'ordonnancement pour des tâches de durée constante

Le résultat obtenu sur les 4 jeux de données montre que la politique SPT-RR est systématiquement moins performante que la politique SPT. Pour les utilisateurs « Data

challenge », les politiques RR et FS ont la même performance tandis que SPT est la meilleure politique. Pour les utilisateurs normaux, la politique FS est moins performante que RR pour le ralentissement maximal mais plus performante pour le ralentissement moyen. SPT reste la meilleure parmi les 5 politiques.

Les figures 5-15 et 5-16 présentent une comparaison des performances de la politique SPT-SPT pour différentes valeurs de p avec les politiques d'ordonnancement les plus performantes (FS, RR et SPT) pour les deux groupes d'utilisateurs (utilisateurs normaux et utilisateurs « data challenge ») pour des tâches de durée constante et variable. Les 4 scénarios ont été considérés (cas 00 à cas 03) avec des contributions croissantes des utilisateurs data challenge.

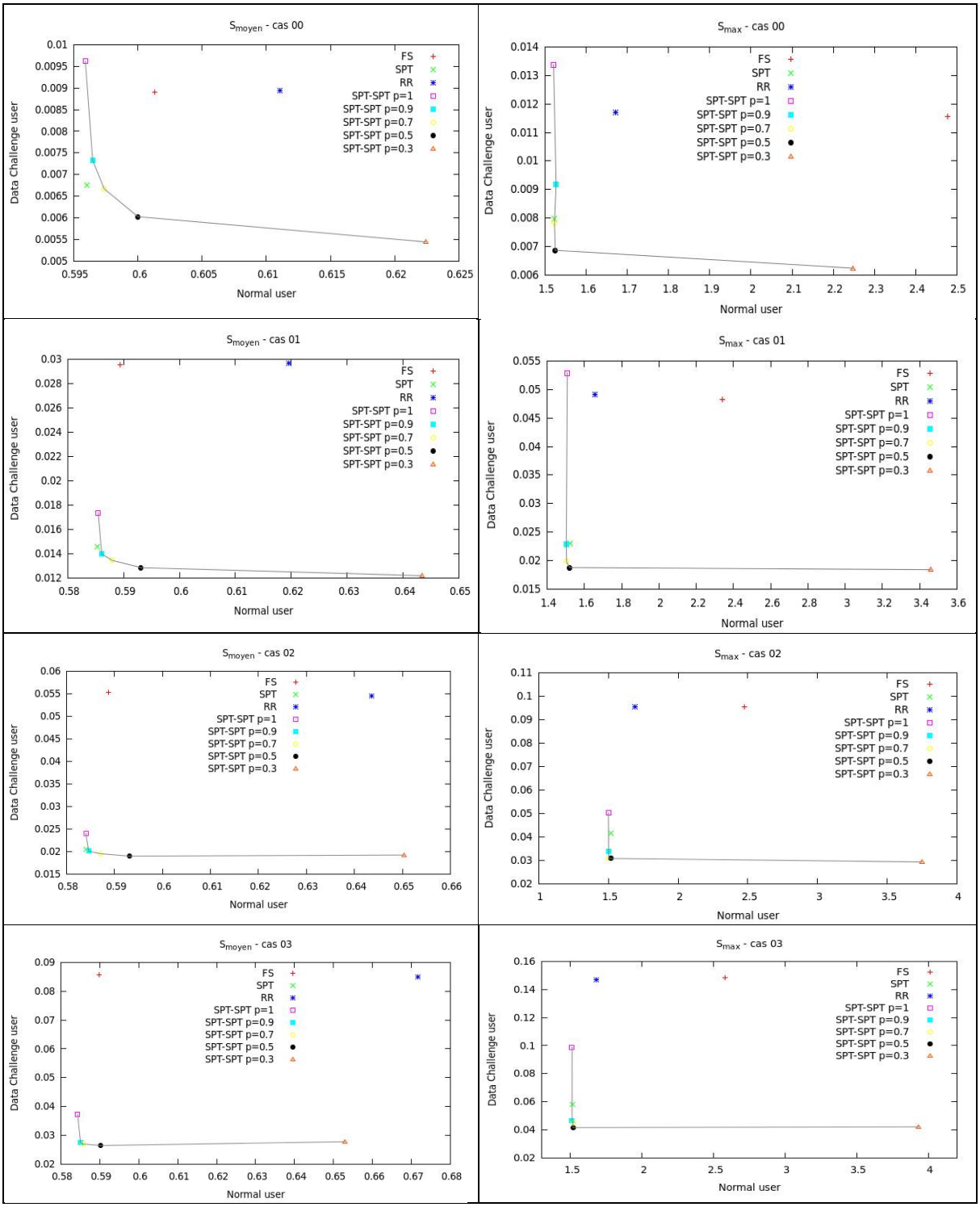


Figure 5-15: Comparaison des performances en termes de ralentissement moyen et maximal de la politique SPT-SPT avec les autres politiques d'ordonnancement pour des tâches de durée variable

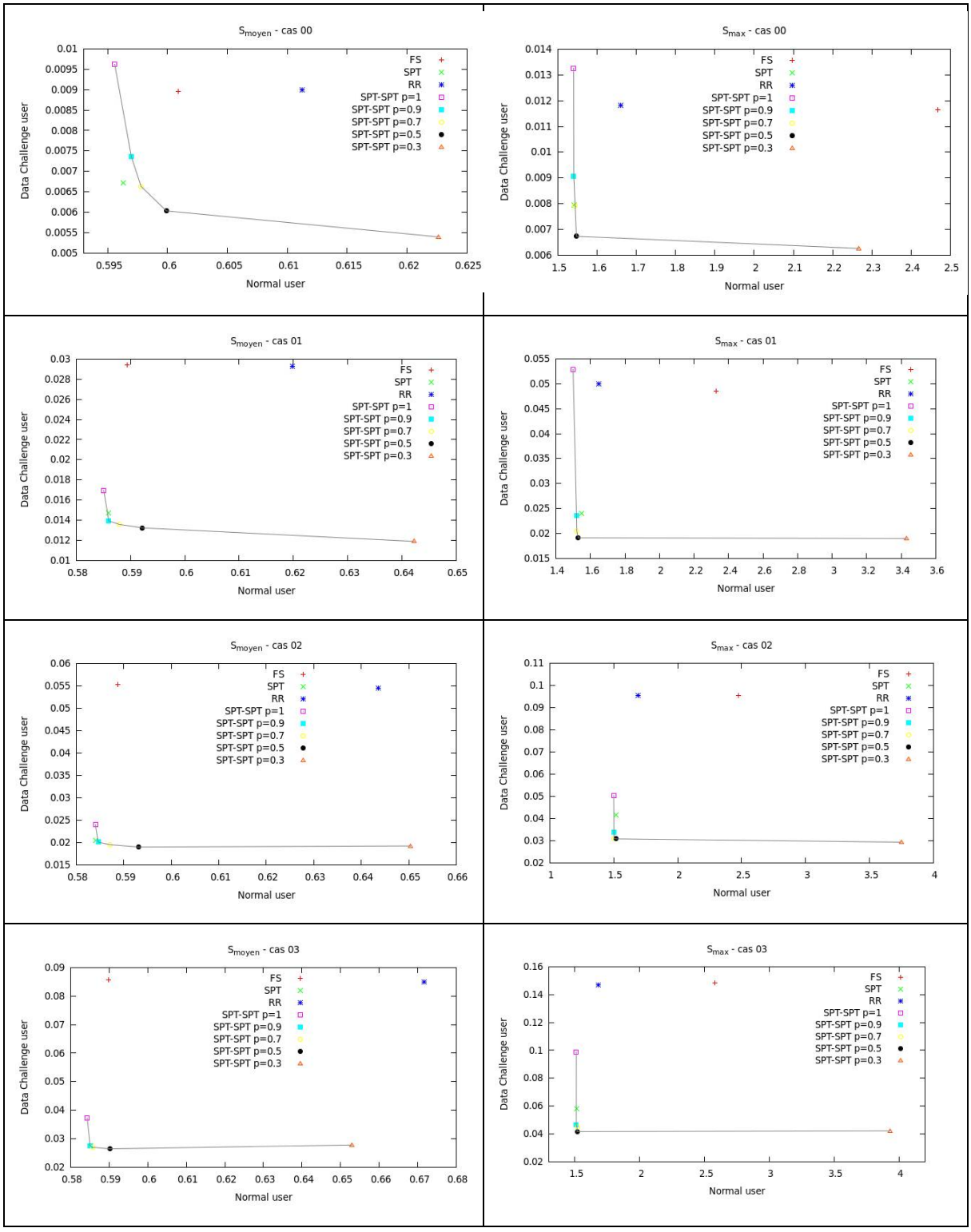


Figure 5-16: Comparaison des performances en termes de ralentissements moyen et maximal de la politique SPT-SPT avec les autres politiques d'ordonnancement pour des tâches de durée constante

Cas 00 et 01 : Par rapport à la politique SPT, le ralentissement moyen pour la politique SPT-SPT avec p = 0,5 diminue pour le groupe « Data Challenge » mais il augmente pour le groupe «Normal». Cependant, le ralentissement maximal de SPT- SPT avec p = 0,5 est significativement diminué pour le groupe « Data Challenge » et n'augmente que légèrement pour le groupe «Normal ».

Cas 02 et 03: Par rapport à la politique SPT, S_{moyen} et S_{max} de SPT-SPT avec p = 0,9 diminuent significativement pour le groupe « Data Challenge » mais augmenté légèrement pour le groupe « Normal ».

Nous constatons donc que, selon la valeur de p, la politique SPT-SPT peut donner de meilleurs résultats que la politique SPT pour le groupe des utilisateurs « Data Challenge »: le ralentissement des utilisateurs « Data Challenge » réduit significativement sans que le ralentissement des utilisateurs normaux soit significativement détérioré. La politique SPT-SPT améliore donc les résultats de la planification par rapport à la politique SPT.

Nous allons maintenant présenter les résultats des politiques d'ordonnancement avec la simulation d'une fédération de clouds.

5.3 Evaluation des politiques d'ordonnancement sur une fédération de clouds

5.3.1 Définition des paramètres de la plate-forme sur la fédération de clouds

Ne disposant pas de traces expérimentales de clouds académiques, nous avons choisi de considérer une fédération de clouds dans la même configuration que l'infrastructure Auvergrid (ref. Table 5-1). Les clusters sont maintenant les « end-point » de la fédération. Au lieu de soumettre des agents pilotes sur grille, la plate-forme va donc soumettre des machines virtuelles sur le cloud fédéré Auvergrid. Lorsqu'une machine virtuelle est créée, elle contacte le gestionnaire de tâche de la plate-forme pour télécharger les tâches à exécuter. S'il n'y a plus de tâche dans la file d'attente, la machine virtuelle va rester active pour une durée définie

par le paramètre TIME_OUT. Après cette période, sans arrivée de nouvelle tâche, la machine virtuelle s'arrête.

Comme nous l'avons vu au paragraphe 4.3.2.7, l'algorithme de contrôle de la création de machine virtuelle de la plate-forme à agents pilotes sur le cloud utilise deux paramètres : CPUperInstance et Iteration_time. Le paramètre CPUPerInstance correspond à la valeur seuil de la somme des durées de calcul des tâches dans la file d'attente. Au-delà de ce seuil, une nouvelle machine virtuelle est soumise. L'ordonnanceur met à jour périodiquement l'état de la file d'attente après une période Iteration_time pour décider de soumettre ou pas une nouvelle machine virtuelle.

5.3.2 Impact des paramètres de la plate-forme

Le choix des paramètres «Iteration_time» et « CPUperInstance » impacte directement le temps total d'utilisation des machines virtuelles ainsi que le temps de finition de toutes les tâches. Pour illustrer cet impact, nous considérons un exemple tiré du jeu de données cas_00 avec seulement 123 utilisateurs normaux soumettant un nombre total de 675 jobs de durée égale à 484,44 secondes, durée moyenne utilisée précédemment. La valeur du paramètre TIME_OUT est fixée à 5 minutes, valeur couramment utilisée pour les images virtuelles VMDIRAC sur EGI. Nous avons testé différentes valeurs des paramètres "Iteration Time" et "CPUperInstance":

- Le temps d'itération prend les valeurs suivantes: 1s, 20s, 40s, 60s, … 450s.
- Le temps minimal de calcul dans la file d'attente, exprimé en unite de tâches de docking (docking/VM), varie de 1 jusqu'à 350 en passant par 50, 100, 150 …

La durée d'utilisation du CPU est le temps total de calcul des machines virtuelles:

$$usage\_time\_CPU = \int number\_VM.dt$$

Nous rappelons aussi la définition du makespan, temps de finition de toutes les tâches :

$$makespan = maxC_i$$

Où C_i est le temps de finition de l'utilisateur i

Dans ce paragraphe, nous allons considérer que l'ordonnanceur de la plate-forme gère les requêtes des utilisateurs avec la politique Round Robin car c'est la politique utilisée dans DIRAC.

Les résultats obtenus pour le makespan et la durée d'utilisation du CPU sont présentés sur les figures 5.17 et 5.18. Pour simplifier la lecture, le temps total d'utilisation sur la figure 5.18 est divisé par 675 x 484,444 = 326997 secondes, durée minimale théorique si tous les processeurs sont de vitesse égale à 1 et sont utilisés en permanence.

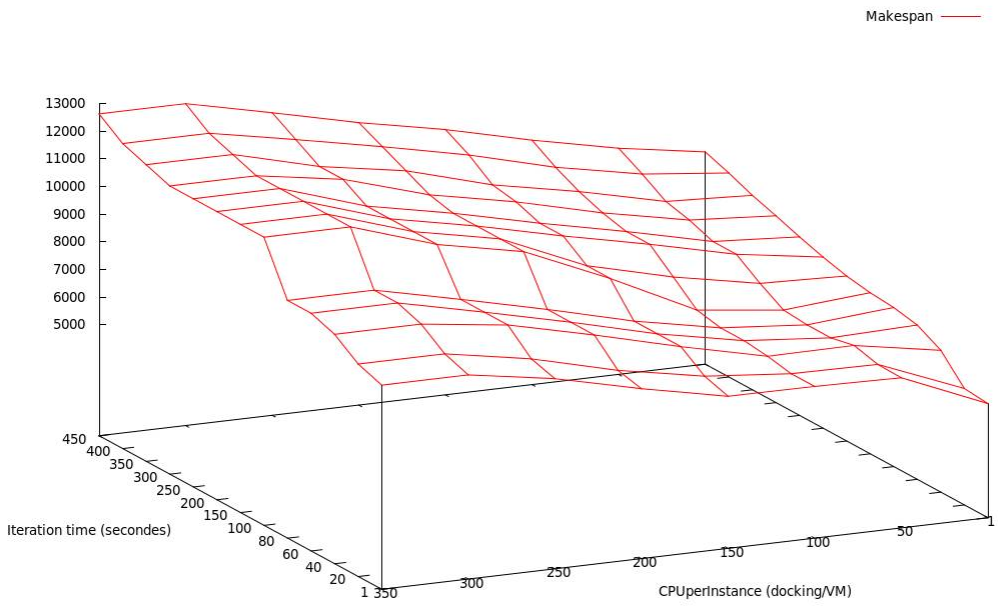


Figure 5-17: Makespan des tâches soumises par les utilisateurs en fonction des paramètres «Iteration_time» et « CPUperInstance »

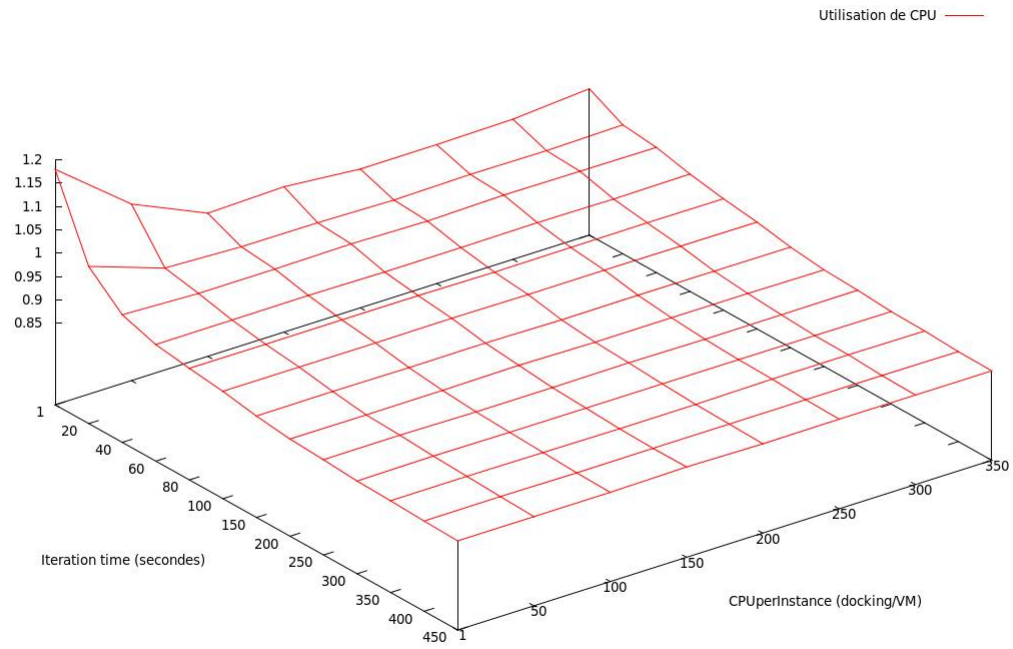


Figure 5-18: Variation de la durée d'utilisation du CPU en fonction des paramètres «Iteration_time» et « CPUperInstance ». La durée totale d'utilisation est divisée par 675 x 484,444 = 326997 secondes, durée minimale théorique si tous les processeurs sont de vitesse égale à 1

Sur la figure 5.17, le meilleur résultat est obtenu lorsque le temps de finition des tâches est minimal, c'est-à-dire quand les paramètres Iteration_time et CPUperInstance sont égaux à 1. Ce temps augmente quand ces paramètres augmentent. En effet, si CPUperInstance et Iteration_time sont égaux à 1, une machine virtuelle est créée dès qu'une tâche arrive dans la file d'attente. Le temps pour créer une nouvelle machine virtuelle est 9 seconds. C'est le temps enregistré sur la plate-forme VMDIRAC à Université Barcelona.

Sur la figure 5.18, nous observons au contraire que, pour ces valeurs, la durée d'utilisation du CPU est maximale. En effet, cette configuration correspond au nombre maximal de machines virtuelles créées et donc une augmentation de l'utilisation du CPU, puisque certaines machines virtuelles resteront actives mais inutilisées. On peut noter sur la figure 5.18 que la durée totale d'utilisation du CPU descend à des valeurs inférieures à 1, mais cela est dû au fait que certains clusters sont équipés de processeurs dont la performance est supérieure à 1 (voir table 5.1).

La figure 5.19 présente la durée totale d'utilisation du CPU en fonction du paramètre TIME_OUT pour un temps d'itération d'une seconde et une valeur de 1 docking/VM pour le paramètre CPUperInstance. Elle illustre le fait que, lorsque TIME_OUT augmente, la

machine reste plus longtemps dans l'attente d'une nouvelle tâche et attend donc une période plus longue avant de s'arrêter, ce qui a pour conséquence une augmentation du temps CPU consommé. La figure 5.20 présente le ralentissement moyen des utilisateurs en fonction du TIME_OUT. Le ralentissement diminue lors que TIME_OUT augmente. En effet, si les machines virtuelles restent plus longtemps disponibles pour les nouvelles tâches qui arrivent, les demandes des utilisateurs sont plus rapidement prises en charge et le ralentissement expérimenté par les utilisateurs diminue.

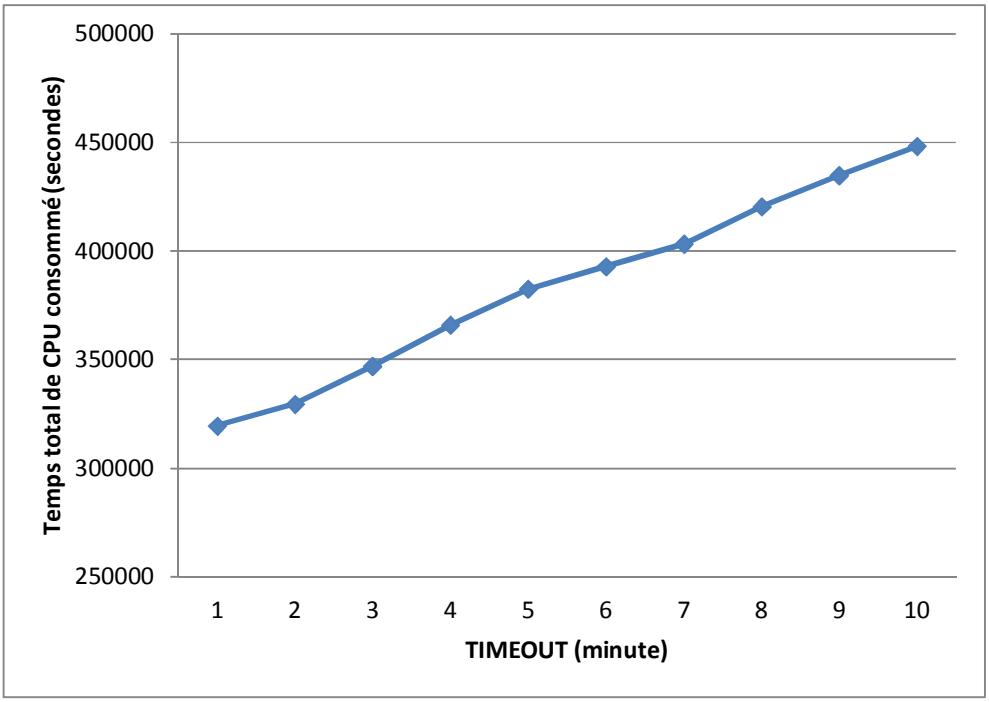


Figure 5-19: Variation de la durée d'utilisation du CPU en fonction du paramètre TIME_OUT

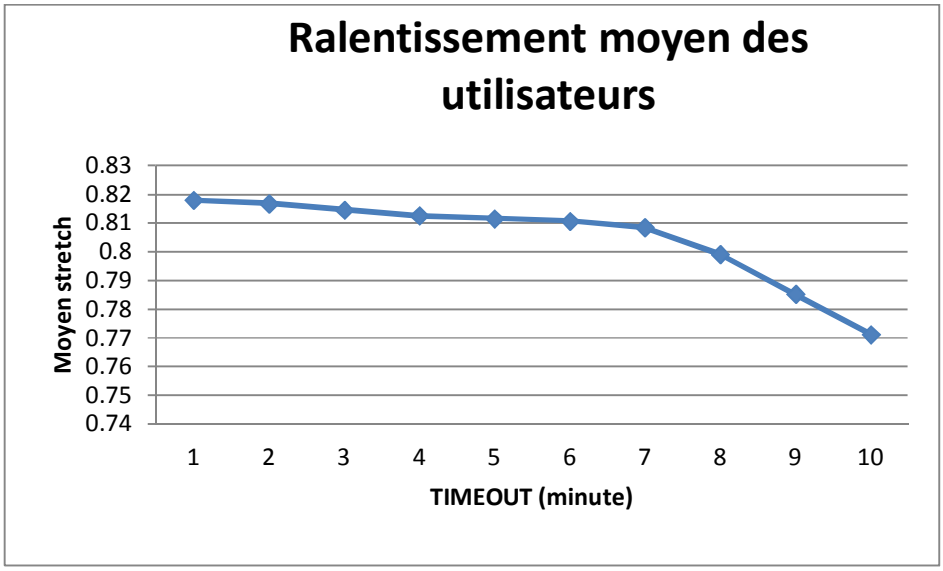


Figure 5-20: Ralentissement des utilisateurs en fonction du paramètre TIME_OUT

Les résultats présentés sur les figures 5.17 à 5.20 ont été obtenus dans un scénario où seuls des utilisateurs normaux ont soumis des tâches. Nous allons maintenant considérer un scénario d'utilisation des ressources de la fédération de clouds via la plate-forme dans un exemple tiré du jeu de données cas_00 mais avec 869 utilisateurs dont 2 utilisateurs « data challenge ». Le scénario décrit la charge de la plate-forme pendant 36000 secondes (10 heures), pendant lesquelles 16528 tâches de durée égale à 484,44 secondes sont soumises, dont 11334 par les 2 utilisateurs « data challenge » et 5194 par les utilisateurs normaux. La politique d'ordonnancement de la plate-forme demeure la politique Round Robin.

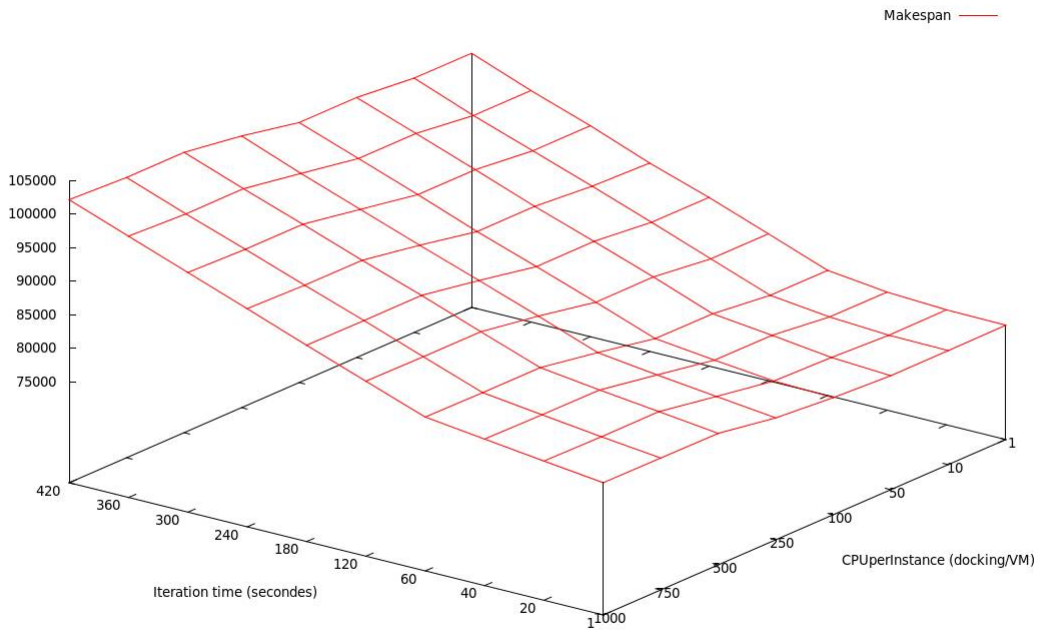


Figure 5-21: : Makespan des tâches soumises par les utilisateurs en fonction des paramètres «Iteration_time» et « CPUperInstance » pour un exemple tiré du jeu de données cas_00 avec 869 utilisateurs dont 2 utilisateurs « data challenge »

Nous retrouvons sur la figure 5.21 le comportement observé pour la figure 5.17 à savoir que le temps de finition des tâches est minimal quand les paramètres Iteration_time et CPUperInstance sont égaux à 1. Ce temps augmente quand ces paramètres augmentent. Le paramètre Iteration_time impacte de façon plus significative le makespan que le paramètre CPUperInstance.

La figure 5.22 décrit la variation de la durée d'utilisation du CPU en fonction des paramètres «Iteration_time» et « CPUperInstance ». La durée totale d'utilisation est divisée par 16.528 x 484,44 = 8.006.824, durée minimale théorique si tous les processeurs sont de vitesse égale à 1 et sont utilisés en permanence.

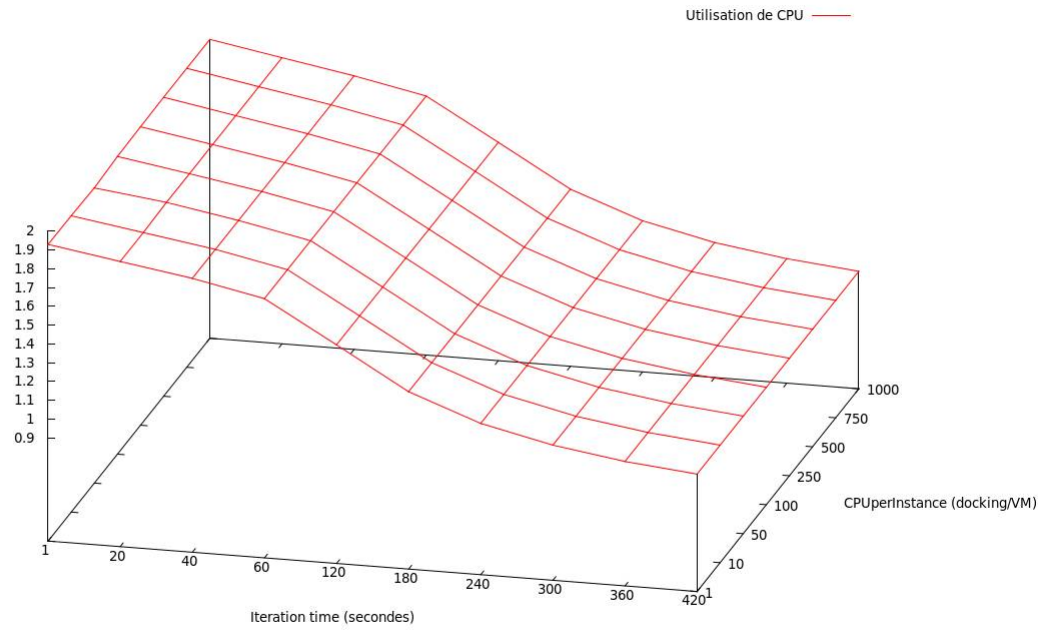


Figure 5-22: Variation de la durée d'utilisation du CPU en fonction des paramètres «Iteration_time» et « CPUperInstance » pour un exemple tiré du jeu de données cas_00 avec 869 utilisateurs dont 2 utilisateurs « data challenge ». La durée totale d'utilisation est d

Nous retrouvons le comportement observé sur la figure 5.18 à savoir que le temps CPU consommé augmente quand Iteration_Time diminue. La plate-forme met à jours l'état de la file d'attente plus souvent, de telle sorte que plus de machines virtuelles sont soumises. Il est aussi à noter que le temps CPU consommé augmente légèrement quand CPUperInstance diminue. Du fait qu'il y a un grand nombre de tâches dans la file d'attente (16.528 tâches soumises au total), la plate-forme soumet de nouvelles machines virtuelles même si CPUperInstance = 10 ou 1000 docking/VM.

Nous nous sommes intéressés aux nombres de machines virtuelles actives en fonction du temps dans ce scénario. La figure 5.23 étudie cette variation pour des valeurs de Iteration_time et CPUperInstance égales respectivement à 60 secondes et 10 docking/VM (figure 5 .23 a) et 60 secondes et 1000 docking/VM (figure 5.23 b)

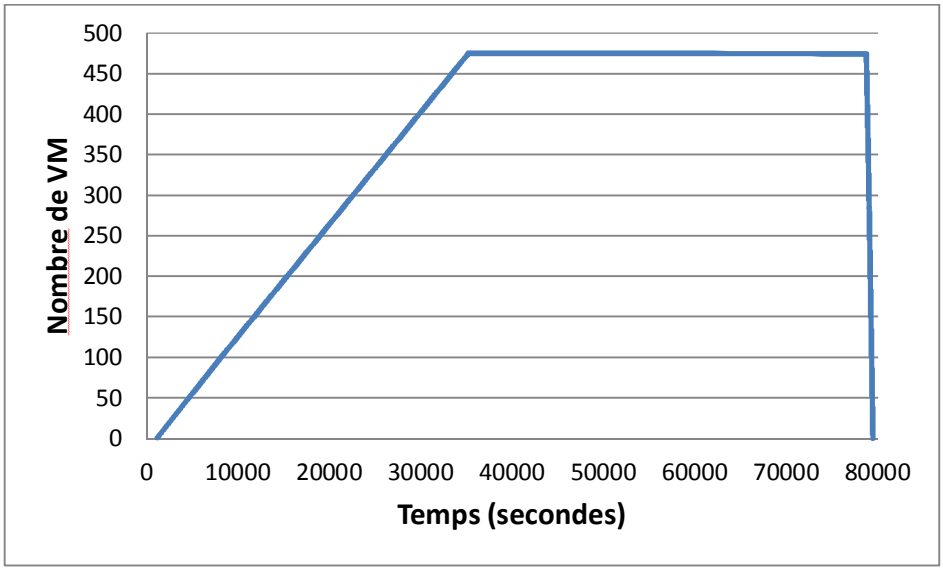


Figure 5-23 a: Nombre de machines virtuelles actives en fonction du temps pour des valeurs de Iteration_time et CPUperInstance égales à 60 secondes et 10 dockings/VM

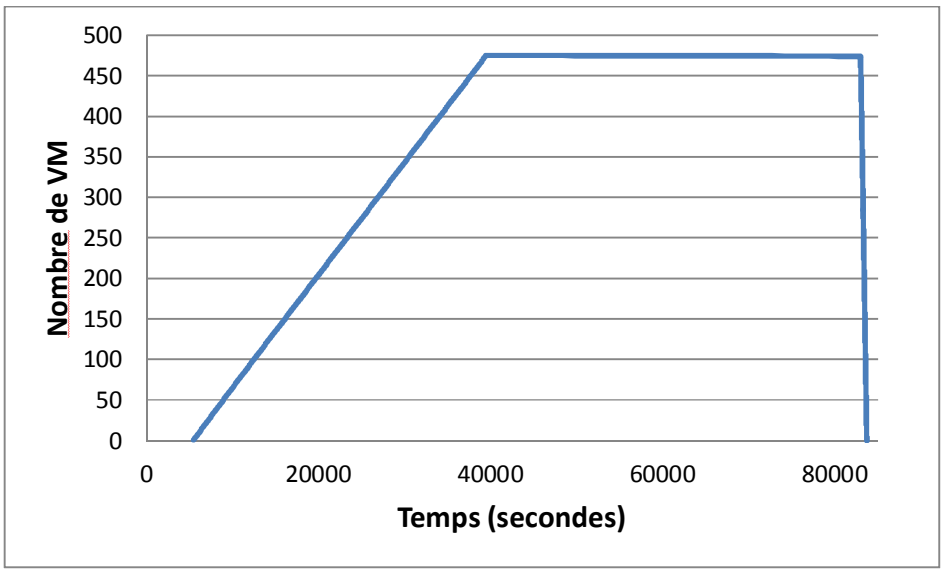


Figure 5.23 b : Nombre de machines virtuelles actives en fonction du temps pour des valeurs de Iteration_time et CPUperInstance égales à 60 secondes et 1000 docking/VM

Les deux figures présentent le même comportement, c'est-à-dire une saturation du nombre de machines virtuelles liée au grand nombre de tâches à accomplir (plus de 16000). Cette saturation est logiquement plus rapide lorsque la valeur du paramètre CPUperInstance est fixée à 10 dockings par VM. Les machines virtuelles sont ensuite utilisées jusqu'à la fin de la simulation parce qu'il y a en permanence des tâches dans la file d'attente.

Pour mieux comprendre l'impact du paramètre Iteration_Time, nous avons représenté sur la figure 5.24 le nombre de machines virtuelles en fonction du temps pour un temps d'itération de 420 secondes, 7 fois supérieur au temps d'itération précédent.

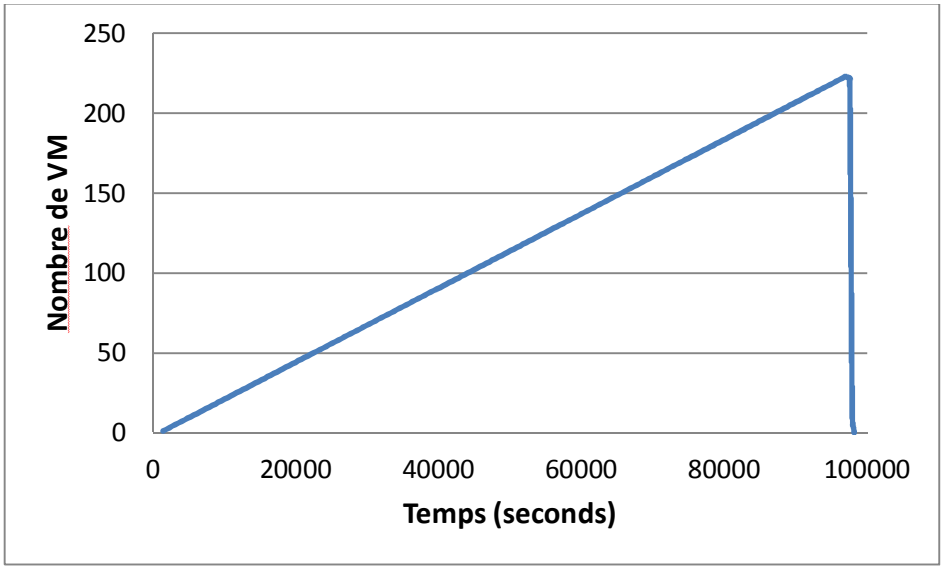


Figure 5-24 a: Nombre de machines virtuelles actives en fonction du temps pour des valeurs de Iteration_time et CPUperInstance égales à 420 secondes et 10 docking/VM

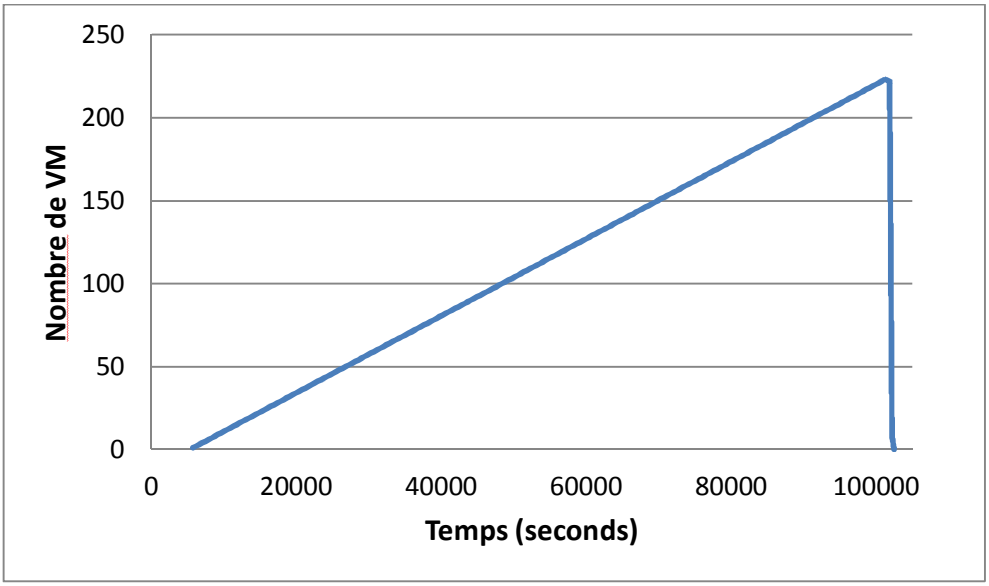


Figure 5.24 b : Nombre de machines virtuelles actives en fonction du temps pour des valeurs de Iteration_time et CPUperInstance égales à 420 secondes et 1000 docking/VM

Sur les figures 5.24a et 5.24b, nous n'observons pas de saturation des ressources proposées par la fédération de cloud.

Si nous regardons maintenant le ralentissement expérimenté par les utilisateurs (figure 5.25), nous observons que celui-ci est dans ce scénario quasi-indépendant du paramètre TIME_OUT

pendant lequel les machines virtuelles attendent de nouvelles tâches avant de s'arrêter. En effet, du fait de la charge importante apportée par les utilisateurs « Data Challenge », les machines virtuelles créées tournent jusqu'à la fin de la simulation.

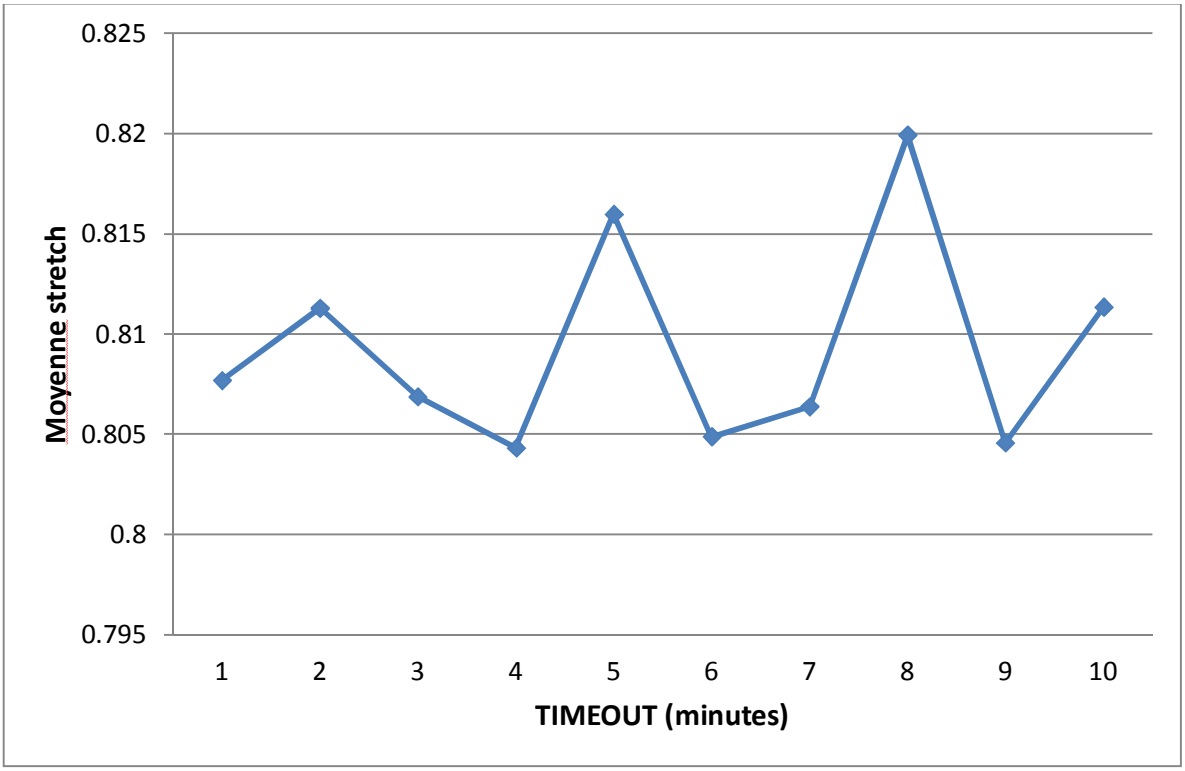


Figure 5-25: Ralentissement des utilisateurs en fonction du paramètre TIME_OUT pour un exemple tiré du jeu de données cas_00 avec 869 utilisateurs dont 2 utilisateurs « data challenge».

En conclusion de ce paragraphe, nous constatons l'importance mais aussi la difficulté d'optimiser les paramètres de la plate-forme, pour qu'il y ait toujours des images virtuelles prêtes lorsque les utilisateurs les demandent afin de réduire l'heure de finition des jobs mais aussi pour minimiser le temps d'attente des machines virtuelles pour réduire le temps d'utilisation du CPU.

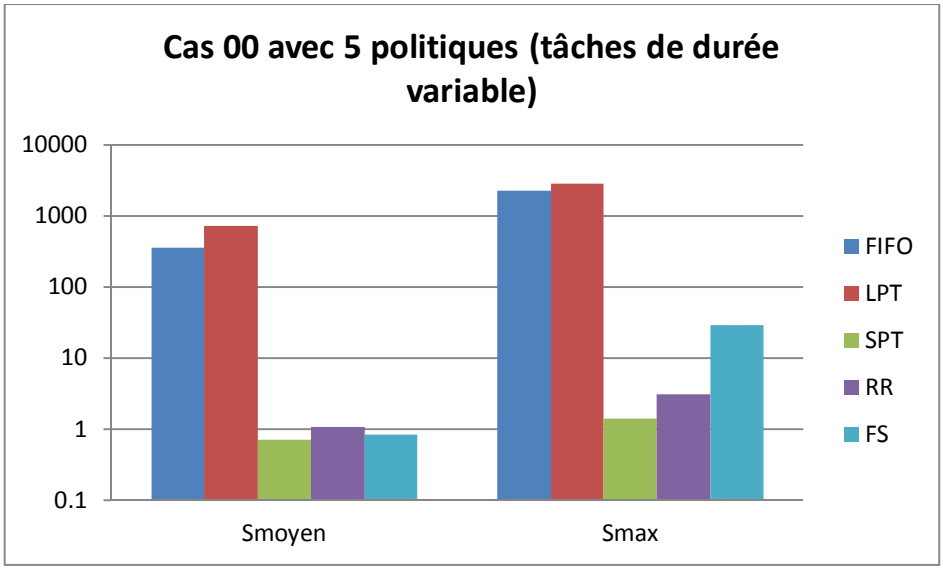
Nous allons maintenant nous intéresser aux politiques d'ordonnancement de la plate-forme pour améliorer au maximum l'expérience des utilisateurs.

5.3.3 Comparaison des politiques d'ordonnancement

Dans ce premier test, nous considérons le cas 00 en changeant la politique d'ordonnancement dans le module Task Manager de la plate-forme d'agent pilote sur la fédération de clouds Auvergrid. Les paramètres de la plate-forme sont les suivants :

- TIME_OUT : 5 minutes
- CPUperInstance : 10 docking / VM
- Iteration_time : 60 secondes

Comme pour la simulation de la grille, nous considérons le cas de tâches de durée constante et de tâches de durée variable. Nous comparons les 5 performances des 5 mêmes politiques d'ordonnancement.



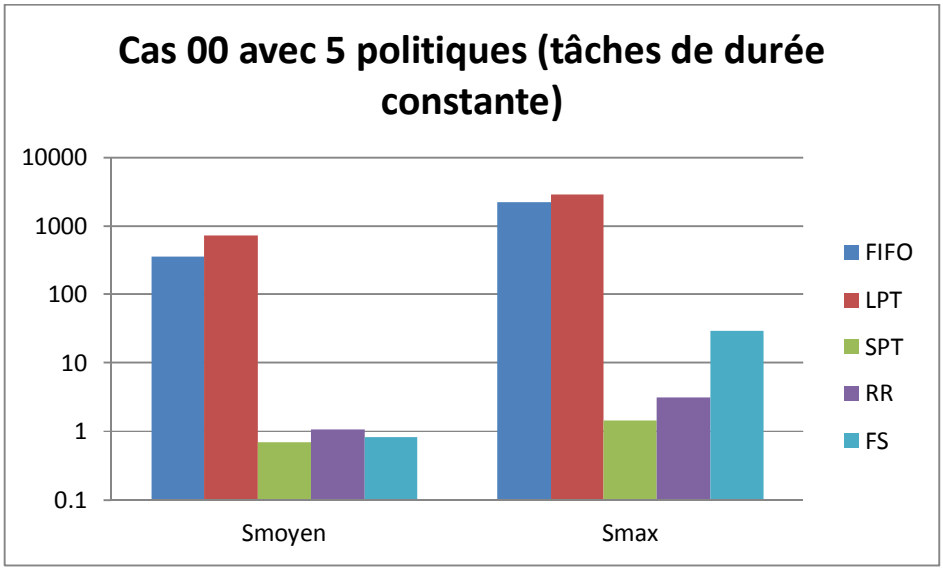


Figure 5-26: Ralentissements moyens et maximaux observés pour 5 politiques d'ordonnancements appliqués sur les tâches soumises à la plate-forme dans le scénario cas_00

Nous retrouvons les résultats observés sur les figures 5.8 et 5.9 : SPT est une très bonne politique d'ordonnancement qui obtient les meilleurs résultats aussi bien pour le ralentissement moyen que pour le ralentissement maximal. Les politiques Fair Share et Round Robin obtiennent des résultats satisfaisants tandis que FIFO et LPT ont des résultats significativement dégradés par rapport aux autres politiques.

S_{moyen}

	FIFO	SPT	LPT	RR	FS
FIFO		500	0	500	500
SPT	0		0	0	0
LPT	500	500		500	500
RR	0	500	0		475
FS	0	500	0	15	

Table 5-8: Comparaison 2 à 2 entre les ralentissements moyens des 5 politiques

S_{max}

	FIFO	SPT	LPT	RR	FS
FIFO		500	37	500	499
SPT	0		0	0	0
LPT	463	500		500	500
RR	0	500	0		0
FS	1	500	0	500	

Table 5-9: Comparaison 2 à 2 entre les ralentissements maximaux des 5 politiques

Sur le modèle des tables 5.4 et 5.5, les tables 5.8 et 5.9 présentent une comparaison 2 à 2 des performances des différentes politiques :

- pour le ralentissement moyen, la politique SPT est la meilleure dans tous les cas. La politique FS est meilleure que RR dans la plupart des cas (475/500).
- Pour le ralentissement maximal, la politique SPT est aussi la meilleure dans tous les 500 cas. La politique RR est aussi meilleure que FS dans tous les cas.

5.3.4 Impact de politiques spécifiques aux groupes d'utilisateurs de la fédération de clouds

Pour pallier à la difficulté d'implémenter la politique SPT en présence d'utilisateurs « data challenge », nous avons proposé dans le paragraphe 5.2.3, deux nouvelles politiques qui utilisent une technique d'ordonnancement différenciée des files d'attente des utilisateurs normaux et « data challenge » pour la planification dans la plate-forme d'agents pilotes. Nous avons appelé SPT-SPT la politique d'ordonnancement dans laquelle les deux groupes d'utilisateurs utilisent la politique SPT et SPT-RR la politique d'ordonnancement dans laquelle le groupe d'utilisateurs normaux utilise la politique SPT et le groupe « Data Challenge » utilise la politique RR. Un paramètre p ($p \in [0,1]$) est introduit pour décrire la probabilité que la file d'attente des tâches des utilisateurs normaux soit choisie pour envoyer ses tâches aux agents pilotes plutôt que la file d'attente des utilisateurs « data challenge ». Par exemple, pour une valeur de p égale à 1, toutes les tâches des utilisateurs normaux seront envoyées avant les tâches des utilisateurs « Data Challenge ».

Nous avons étudié les performances de ces deux nouvelles politiques sur la fédération de clouds AuverGrid pour les 500 exemples des 4 scénarios cas_00 à cas_03. Pour rappel, ceux-ci diffèrent par le nombre d'utilisateurs data challenge et leur poids dans le nombre de jobs soumis (voir table 5-3). Ce poids est croissant de cas_00 à cas_03.

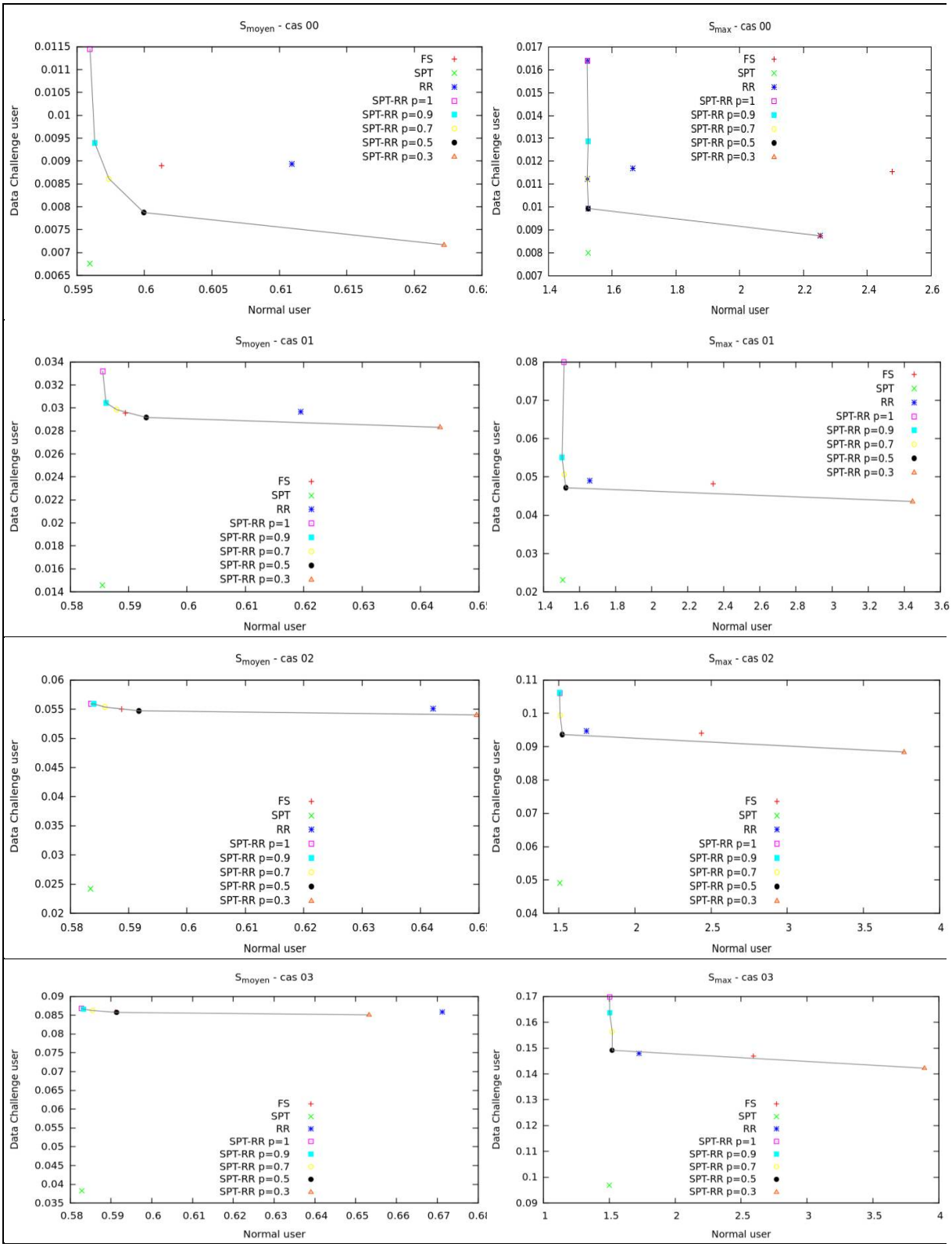


Figure 5-27: Comparaison des performances en termes de ralentissements moyen et maximal de la politique SPT-RR avec les autres politiques d'ordonnancement pour des tâches de durée variable

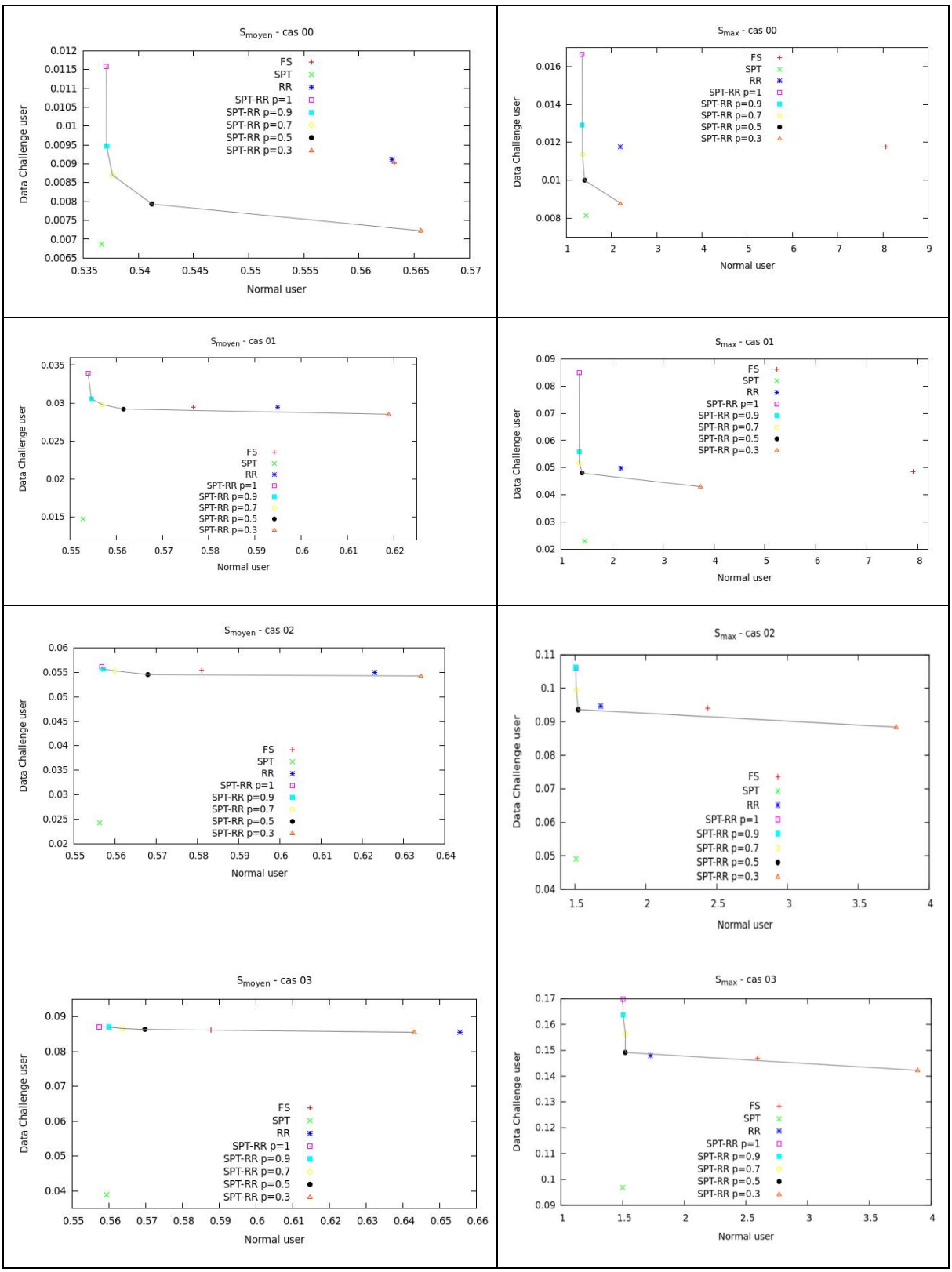


Figure 5-28: Comparaison des performances en termes de ralentissements moyen et maximal de la politique SPT-RR avec les autres politiques d'ordonnancement pour des tâches de durée constante

De façon cohérente, les figures 5.27 et 5.28 montrent que la politique SPT-RR est systématiquement moins bonne que la politique SPT. Le ralentissement expérimenté par les utilisateurs normaux et « Data Challenge » est toujours plus grand que celui expérimenté avec la politique SPT.

Pour les utilisateurs “Data Challenge”, les politiques Round Robin et FairShare ont les mêmes performances. Pour les utilisateurs normaux, la politique FairShare est moins bonne que la politique Round Robin pour le ralentissement maximal mais la politique FairShare est meilleure que Round Robin pour le ralentissement moyen dans les cas 01, 02, 03. Les politiques FairShare et Round Robin ont la même performance dans le cas 00.

Les figures 5-29 et 5-30 présentent les ralentissements moyens et maximaux obtenus pour les groupes normaux et « data challenge » pour la politique SPT-SPT comparée aux politiques SPT, FS et RR pour les mêmes cas.

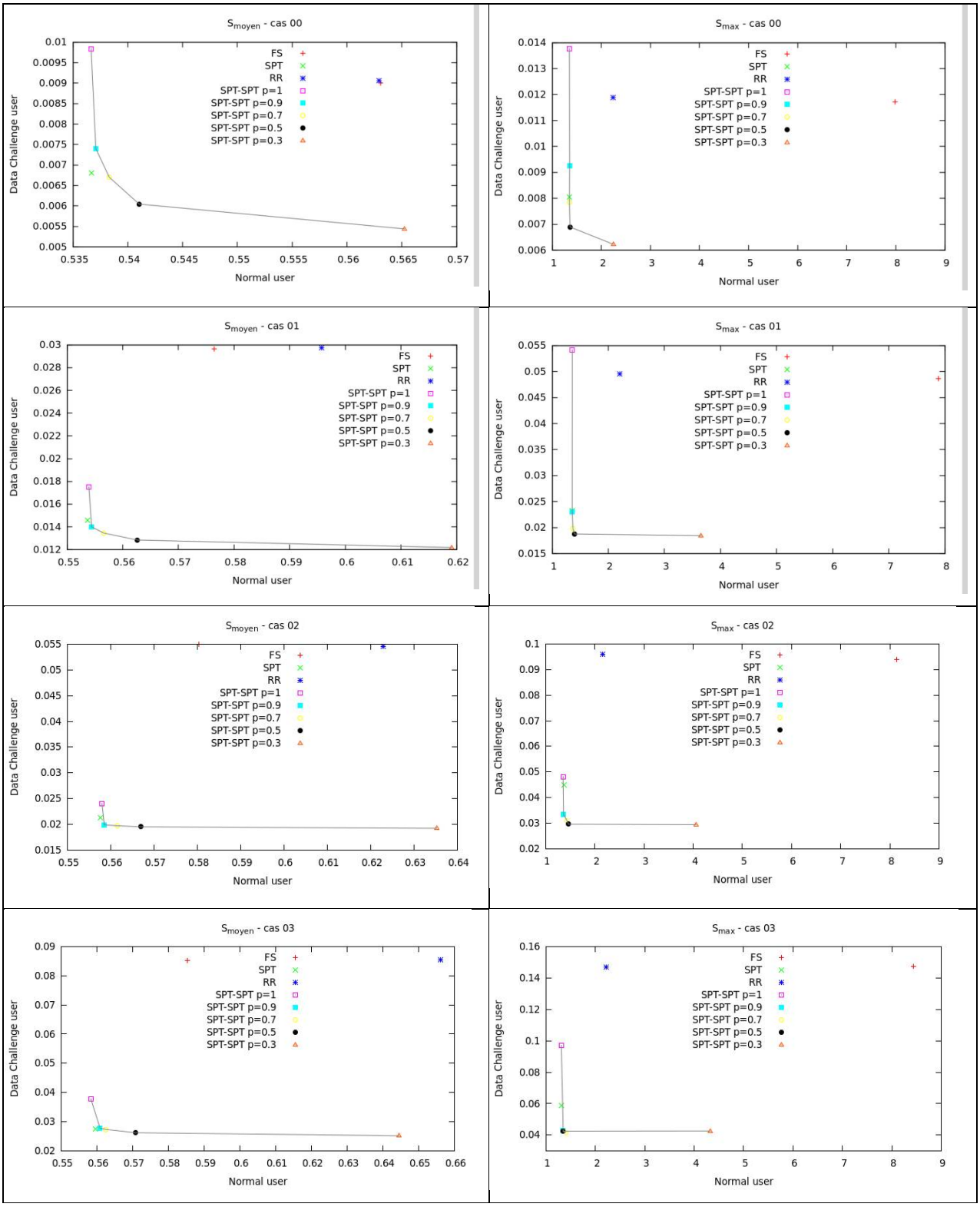


Figure 5-29: Comparaison des performances en termes de ralentissements moyen et maximal de la politique SPT-SPT avec les autres politiques d'ordonnancement pour des tâches de durée variable

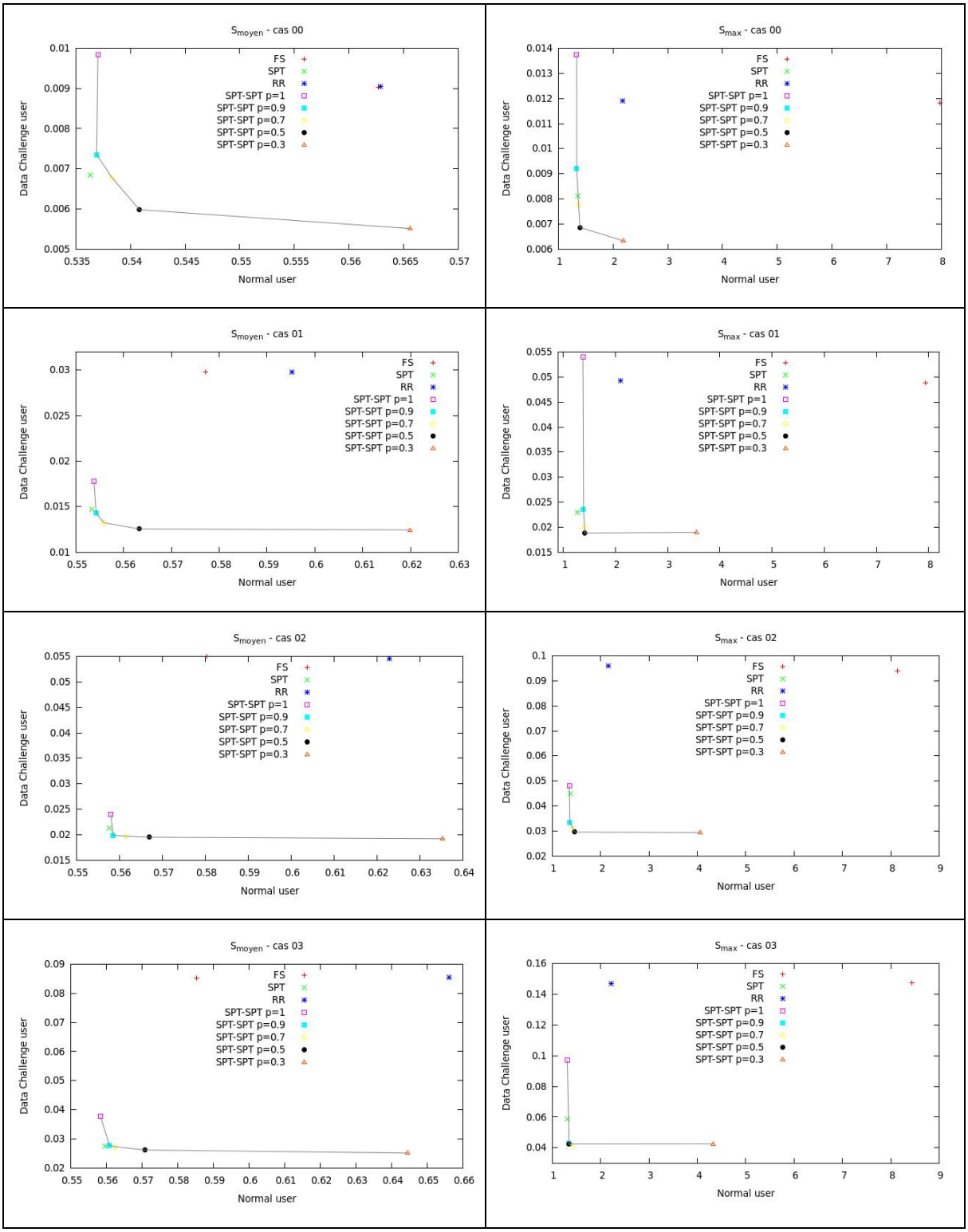


Figure 5-30: Comparaison des performances en termes de ralentissements moyen et maximal de la politique SPT-SPT avec les autres politiques d'ordonnancement pour des tâches de durée constante

Dans les scénario cas_00 et cas_01, les utilisateurs « data challenge » expérimentent des ralentissements moyens et maximaux plus faibles avec la politique SPT-SPT par rapport à la politique SPT pour des valeurs de p inférieures à 0,5, c'est-à-dire lorsque la probabilité de sélectionner leur file d'attente, égale à (1-p) est supérieure à 50%. Cette amélioration se fait bien sûr aux dépens des utilisateurs normaux.

Dans les scénarios cas_02 et cas_03, le phénomène est accentué avec le poids croissant des utilisateurs « Data Challenge ». Avec le paramètre p = 0.9, la politique SPT-SPT est meilleure que SPT.

Les résultats sont très homogènes selon que les tâches considérées sont de durée constante ou variable.

5.4 Conclusion

Nous avons consacré ce chapitre à la description des expériences de simulation menées avec SimGrid et à l'analyse des résultats obtenus. Nous avons considéré une même infrastructure physique, celle de la grille régionale Auvergrid, dont les ressources sont utilisées par la plate-forme à agents pilotes soit en configuration de grille, soit en configuration de clouds. Nous avons retrouvé dans les deux configurations la supériorité de la politique d'ordonnancement SPT (Smallest Processing Time) pour optimiser les ralentissements expérimentés par les utilisateurs.

Pour améliorer l'expérience des plus gros utilisateurs, pénalisés par cette politique qui privilégie systématiquement les tâches les plus courtes, nous avons proposé de l'appliquer à deux files d'attente séparées, celle des utilisateurs dits normaux et celle des utilisateurs avec beaucoup de tâches. Les résultats montrent qu'il est possible d'améliorer le ralentissement expérimenté par les utilisateurs avec beaucoup de tâches sans pénaliser significativement les utilisateurs normaux.

Dans le chapitre suivant, nous allons confronter nos observations à la réalité expérimentale d'infrastructures opérationnelles.

CHAPITRE 6 : RESULTATS EXPERIMENTAUX SUR LA GRILLE EGI ET LE CLUSTER DE KISTI

Dans le chapitre précédent, nous avons présenté les résultats de simulations déployées à l'aide de SIMGRID pour choisir la meilleure stratégie d'ordonnancement des tâches de criblage virtuel soumises à une plate-forme à agents pilotes sur des infrastructures de grille et de cloud. La prochaine étape est de confronter nos observations à des tests expérimentaux sur des infrastructures opérationnelles pertinentes dans la perspective d'un travail depuis le Vietnam. Nous avons considéré pour cela d'une part l'infrastructure de grille EGI et d'autre part un des clusters de PLSI (Partnership & Leadership for the nationwide Supercomputing Infrastructure), infrastructure distribuée de supercalculateurs en Corée du Sud.

Les résultats présentés dans ce chapitre ont été obtenus avec les plates-formes à agents pilotes DIRAC et HTCaaS. Pour comparer la politique SPT à l'algorithme d'ordonnancement original de la plate-forme, nous avons considéré un scénario avec 2 utilisateurs sur DIRAC et HTCaaS. Pour comparer entre SPT et SPT-SPT, nous avons considéré un scénario à 1000 utilisateurs sur HTCaaS. Le résultat obtenu montre que la politique SPT est meilleure que la politique originale de la plate-forme, et la politique SPT-SPT est meilleure que la politique SPT pour optimiser le ralentissement des utilisateurs.

6.1 Expérimentation avec DIRAC sur EGI

Nous avons vu dans le paragraphe 2.2.3 que l'infrastructure européenne EGI accueillait des communautés d'utilisateurs en sciences du vivant, notamment dans l'organisation virtuelle Biomed. Nous avons aussi décrit la plate-forme DIRAC [43] au paragraphe 2.4.5.

Nous allons présenter dans un premier temps la méthode que nous avons utilisée pour implémenter de nouvelles politiques d'ordonnancement dans DIRAC. Nous présenterons ensuite un calcul des ralentissements attendus avec la politique RR et la politique SPT dans le test déployé sur EGI puis nous discuterons les résultats obtenus.

6.1.1 Implémentation des politiques d'ordonnancement

La figure 6.1 présente les différents modules de la plate-forme DIRAC.

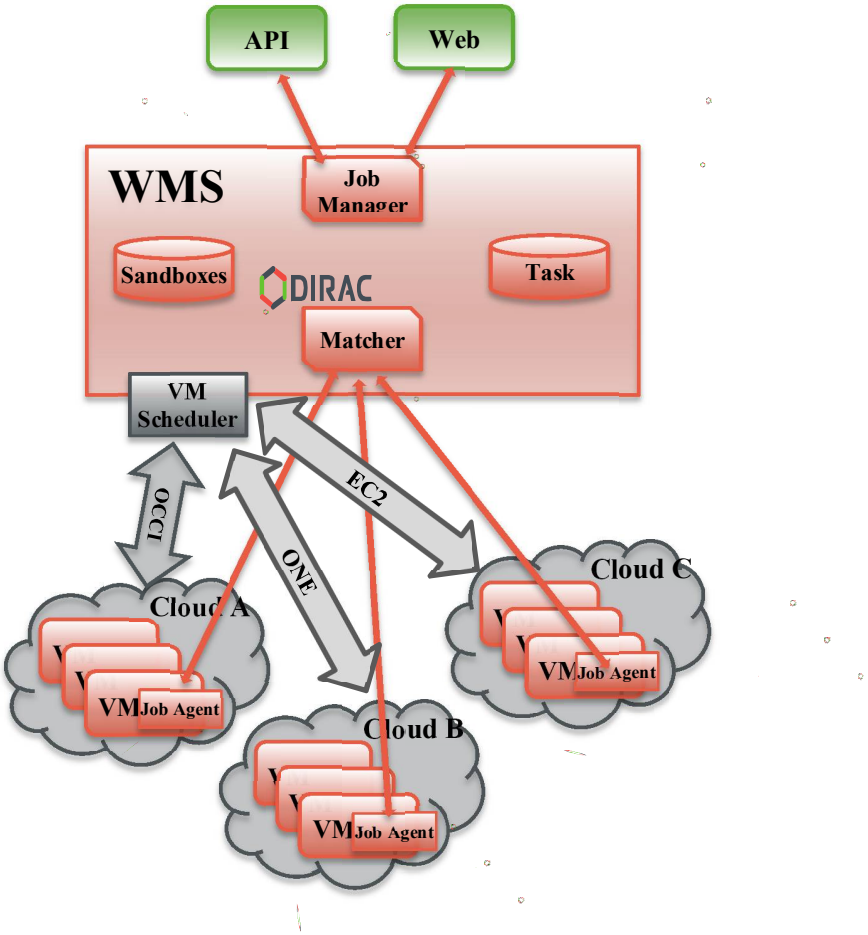


Figure 6-1: Représentation schématique des différents modules de DIRAC.

Quand DIRAC reçoit des tâches soumises par des utilisateurs, elles sont stockées dans une queue (TaskQueue) et une priorité est assignée à ces tâches. Ensuite lorsque DIRAC reçoit une demande d'un agent pilote, le module « Matcher » envoie la tâche qui a la plus haute priorité. Un module « Corrector » met à jour la priorité de chaque tâche dans la queue. Dans le cas d'un déploiement sur un cloud ou une fédération de clouds, c'est le module VM Scheduler qui déploie les images virtuelles VM DIRAC.

De cette façon, DIRAC permet aux utilisateurs de définir une politique de planification spécifique en programmant son propre module « Corrector ». Nous avons donc installé un serveur DIRAC au Laboratoire de Physique Corpusculaire de Clermont-Ferrand. Sur ce

serveur, nous avons créé un groupe nommé « docking_usr » d'utilisateurs de la VO biomed. Ensuite, nous avons programmé le module « Corrector » selon la politique SPT (avec l'aide d'Andrei Tsaregorodtsev - Centre de Physique des Particules de Marseille sur la programmation dans la plate-forme DIRAC). La modification se fait en deux étapes.

Etape 1: Implémentation du module « Corrector » modifié et mise à jour du répertoire de DIRAC

La plate-forme DIRAC est écrite en Python. Pour ajouter un nouveau module « Corrector », nous programmons le nouvel algorithme en Python pour mettre à jour la priorité des tâches des utilisateurs dans le module TaskQueue de DIRAC. Ensuite ce fichier est introduit dans le répertoire $DIRAC_Root/WorkloadManagementSystem/private/correctors/. Pour la politique SPT, nous avons créé un fichier SPTCorrector.py, sauvegardé à l'adresse $DIRAC_Root/WorkloadManagementSystem/private/correctors/SPTCorrector.py.

Etape 2: Mise à jour de la configuration de DIRAC

Une fois le nouveau groupe « docking_usr » d'utilisateurs de la VO Biomed créé, nous modifions la configuration de DIRAC pour qu'il reconnaisse notre fichier « SPTCorrector » ajouté.

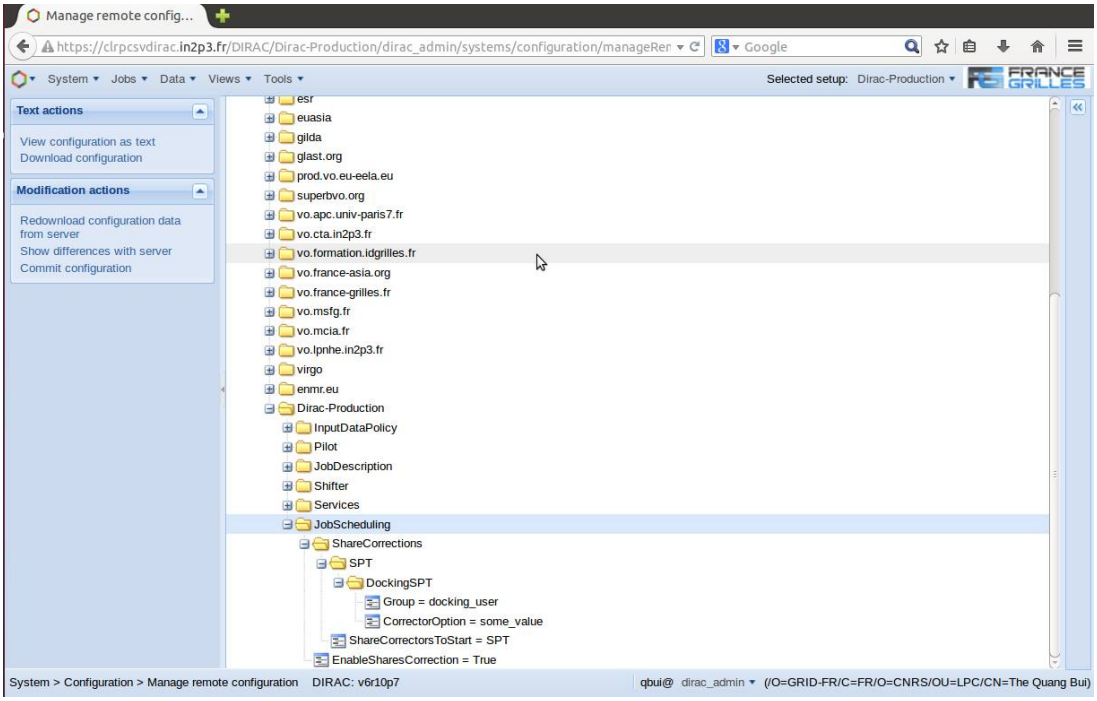


Figure 6-2: Configuration de DIRAC

Les champs ajoutés sont les suivants (figure 6-2):

/Operations/Dirac-Production/JobScheduling/ShareCorrections/SPT/DockingSPT

- Group=docking_user
- CorrectorOption=<some_value>

/Operations/Dirac-Production/JobScheduling/ShareCorrections/SPT

- ShareCorrectorsToStart=SPT

/Operations/Dirac-Production/JobScheduling/ShareCorrections/

- EnableSharesCorrection=True

Après installation du code source de notre “Corrector” dans le répertoire de DIRAC et la mise à jour de la configuration de DIRAC, le serveur DIRAC va exécuter la politique SPT dans le groupe docking_user.

6.1.2 Scénario d’expérimentation

Nous utilisons un scénario à 2 utilisateurs de l’organisation virtuelle Biomed dans lequel l’utilisateur 1 soumet 150 tâches, l’utilisateur 2 200 tâches de même durée et l’utilisateur 2

arrive juste après l'utilisateur 1. Les tâches soumises par l'utilisateur dans ce test sont le même docking entre le ligand ZINC00001967 dans la base de donnée ZINC et la protéine 1EVE – la cible biologique de la maladie d'Alzeihmer. La durée de ce docking est 484.28 seconds sur le CPU Intel Duo Core 2 Ghz. Le serveur DIRAC installé au Laboratoire de Physique Corpusculaire a été utilisé pour ce test (http://clrpcsvdirac.in2p3.fr).

L'ordonnanceur original de DIRAC répartit équitablement la ressource entre les utilisateurs dans un groupe, ce qui correspond à la politique Round Robin (RR).

Nous pouvons calculer le ralentissement théorique expérimenté dans ce scénario par les utilisateurs avec les politiques SPT et RR.

Considérons pour cela qu'un nombre M de CPUs sont disponibles pour ce test et que ces CPUs ont la même vitesse. Le temps pour finir une tâche sur un CPU est P. Le temps de réponse pour un utilisateur est la durée entre le moment de soumission de ses tâches et celui de finition de toutes ses tâches.

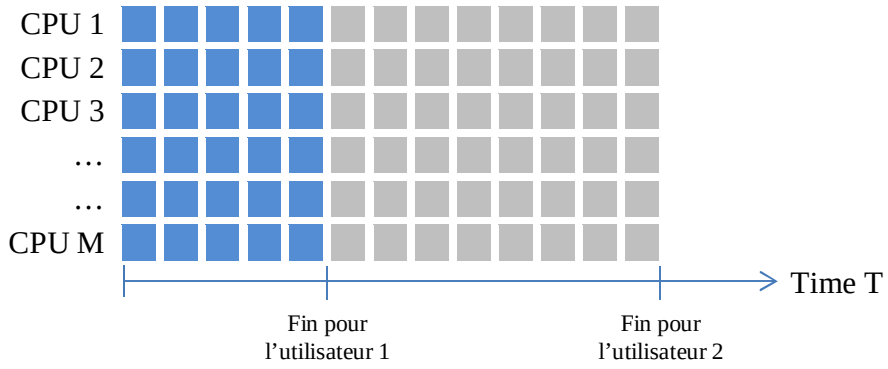


Figure 6-3: Utilisation de CPU dans la politique SPT

La figure 6-3 représente l'utilisation des ressources avec la politique SPT. Dans un premier temps, l'utilisateur 1 prend tous les CPUs parce qu'il a soumis moins de tâches. Quand l'exécution des tâches de l'utilisateur 1 est terminée, l'utilisateur 2 prend à son tour tous les CPUs.

On a alors:

Temps de réponse de l'utilisateur 1 : 150*P/M

Durée d'exécution des tâches de l'utilisateur 1 : 150*P

⇨ Ralentissement de l'utilisateur 1 : 1/M

Temps de réponse pour l'exécution des tâches de l'utilisateur 2 : (200 + 150)*P/M

Durée d'exécution des tâches de l'utilisateur 2 : 200*P

⇨ Ralentissement de l'utilisateur 2 : 350/(200 * M)

La figure 6-4 représente l'utilisation du CPU dans le cas de la politique RR.

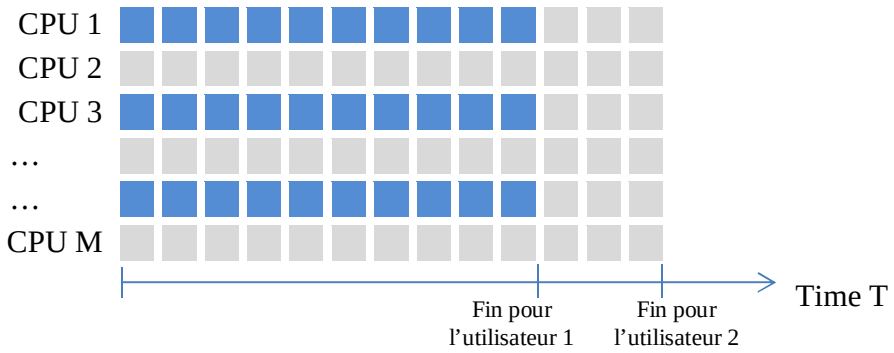


Figure 6-4: Utilisation du CPU dans la politique RR

Tout d'abord les deux utilisateurs partagent les CPUs, chaque utilisateur prend la moitié du nombre de CPUs. L'utilisateur 1 ayant soumis moins de tâches que l'utilisateur 2, leur exécution se finit en premier et l'utilisateur 2 prend alors tous les CPUs.

On a alors:
Temps de réponse de l'utilisateur 1: (150*2)*P/M
Durée d'exécution des tâches de l'utilisateur 1: 150*P

⇨ ralentissement de l'utilisateur 1 = 2/M

Temps de réponse de l'utilisateur 2 : (150+200)*P/M
Durée d'exécution des tâches de l'utilisateur 2 : 200*P

⇨ ralentissement de l'utilisateur 2 = 350/(200*M)

Ainsi le résultat de ce calcul théorique est le suivant :

- Le ralentissement de l'utilisateur 1 avec la politique SPT est égal à 1/M, soit la moitié du stretch dans le cas RR (2/M)

- Le stretch de l'utilisateur 2 est le même avec les politiques SPT et RR (350 / (200*M))

Nous allons maintenant confronter ces prédictions aux résultats expérimentaux.

6.1.3 Résultat expérimental

Nous avons fait tourner 500 fois le scénario décrit ci-dessus sur la grille EGI. Le ralentissement de chaque utilisateur a été enregistré à chaque fois et nous avons ajusté par un lissage gaussien l'histogramme du ralentissement de chaque utilisateur pour corriger le résultat de l'effet liè à la charge de la grille.

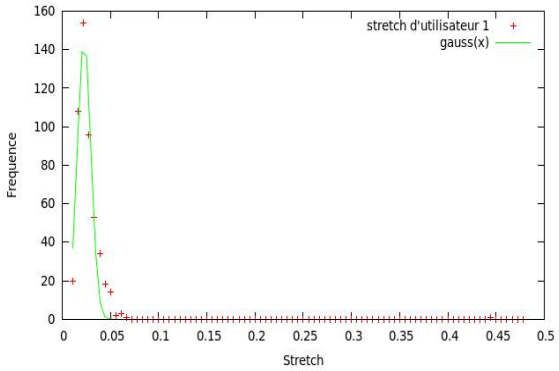


Figure 6-5: : Histogramme du ralentissement de l'utilisateur 1 avec la politique SPT

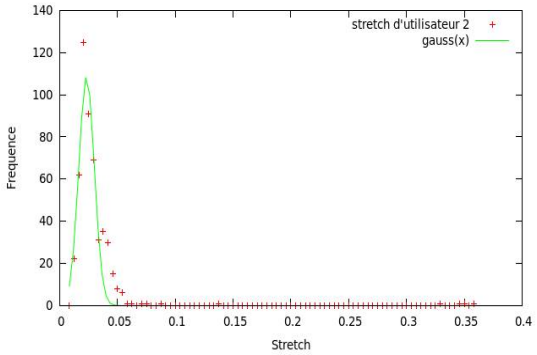


Figure 6-6: Histogramme du ralentissement de l'utilisateur 2 avec la politique SPT

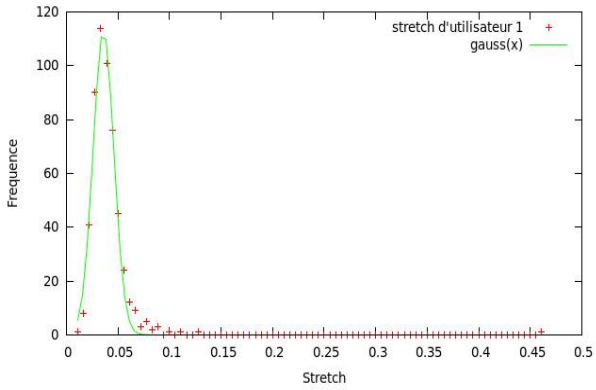


Figure 6-7:Histogramme du ralentissement de l'utilisateur 1 avec la politique RR

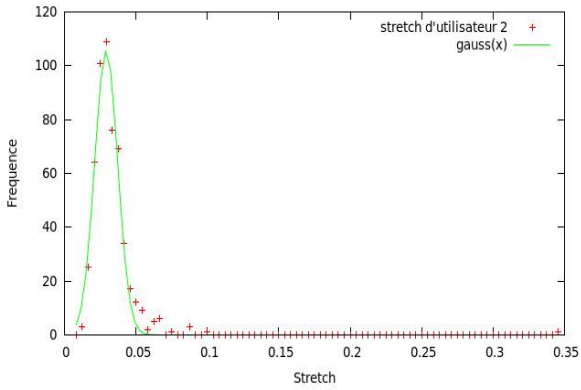


Figure 6-8: Histogramme du ralentissement de l'utilisateur 2 avec la politique RR

La figure 6-5, 6-6, 6-7 et 6-8 représentent l'histogramme du ralentissement des utilisateurs 1 et 2 avec la politique SPT et l'ordonnanceur original de DIRAC. Après l'application du lissage gaussien, nous obtenons les valeurs suivantes pour le ralentissement moyen des utilisateurs (table 6-1 et figure 6-9) :

	Utilisateur 1	Utilisateur 2
L'ordonnancement de DIRAC	0.0359	0.0292
SPT	0.0227	0.0262

Table 6-1: Ralentissement moyen des utilisateurs sur la grille EGI

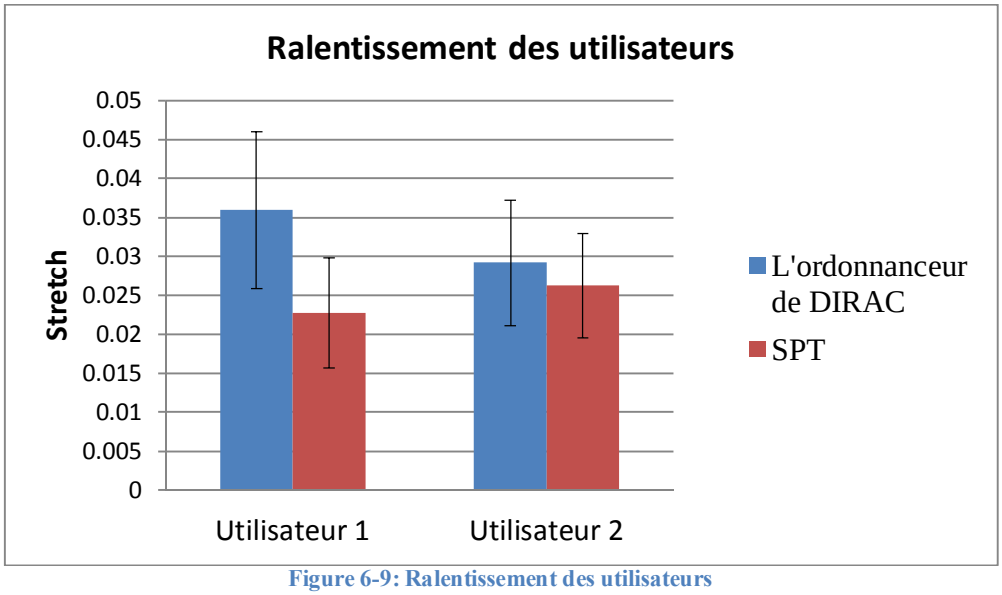


Figure 6-9: Ralentissement des utilisateurs

La figure 6-9 représente le ralentissement moyen des utilisateurs dans le cas de SPT et l'ordonnanceur de DIRAC : les barres d'erreur correspondent à la largeur de la gaussienne. Les ralentissements observés avec la politique SPT sont inférieurs aux ralentissements obtenus dans le cas de l'ordonnanceur de DIRAC.

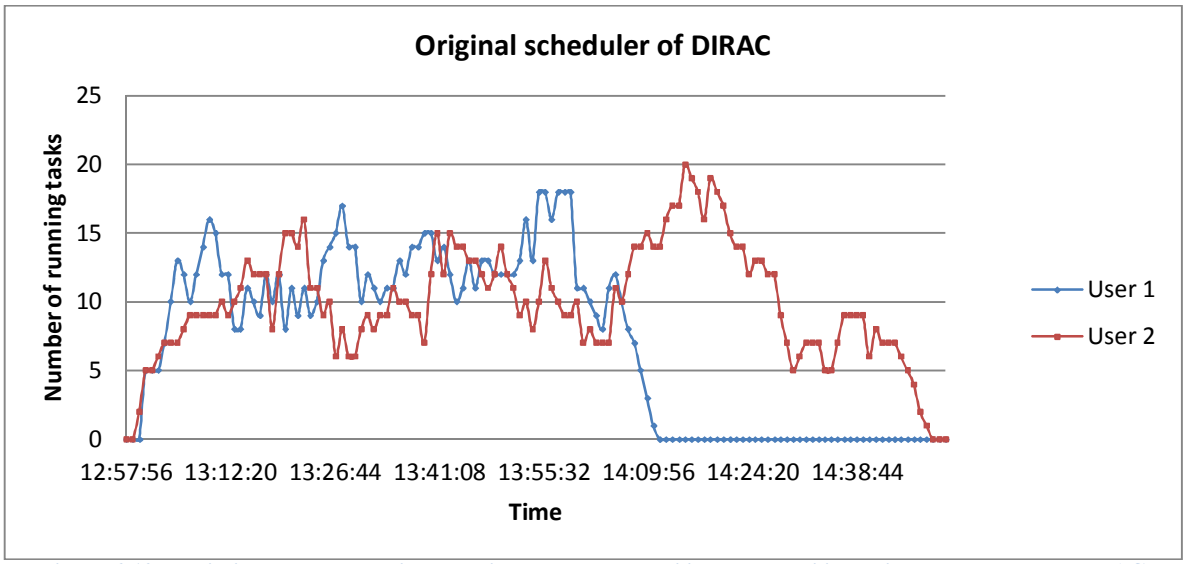


Figure 6-10: Variation du nombre d'agents pilotes de chaque utilisateur en utilisant l'ordonnanceur de DIRAC

La figure 6-10 représente la variation du nombre d'agents pilotes pour chaque utilisateur avec l'ordonnanceur de DIRAC. Au début, les deux utilisateurs partagent de façon équitable le nombre d'agent pilotes disponibles, en cohérence avec la politique RR.

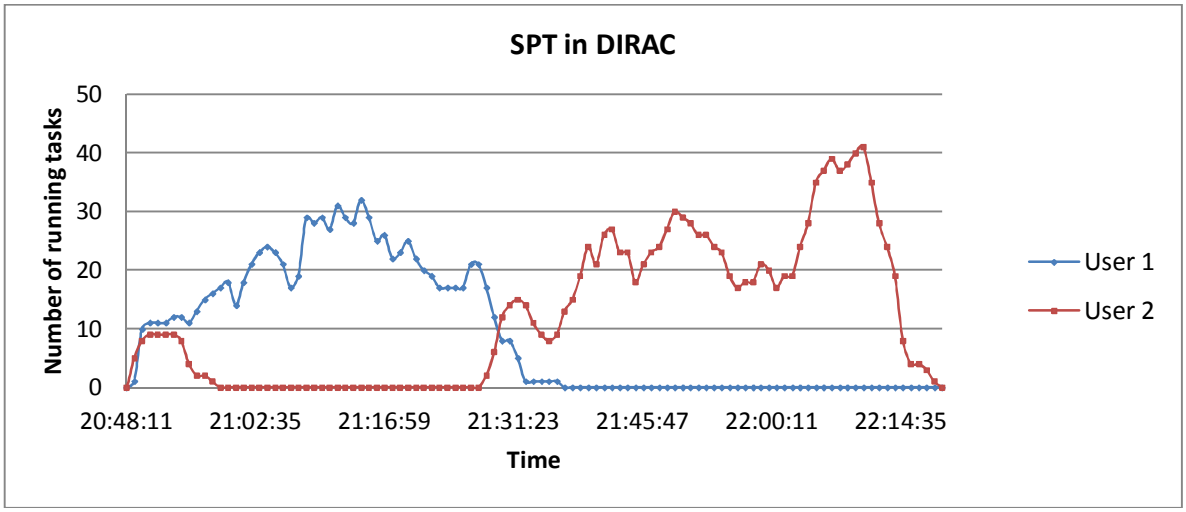


Figure 6-11: Variation du nombre d'agents pilotes de chaque utilisateur avec la politique SPT

La figure 6-11 représente la variation du nombre d'agents pilotes pour chaque utilisateur avec l'ordonnanceur SPT. L'utilisateur 1 prend tous les agents pilotes. Lorsque ses tâches sont finies, l'utilisateur 2 prend à son tour tous les agents pilotes disponibles.

Si nous comparons nos résultats expérimentaux avec le calcul théorique dans le paragraphe précédent, nous n'observons pas le facteur 2 entre les ralentissements expérimentés par l'utilisateur 1 suivant la politique d'ordonnancement. L'origine de cette différence avec le calcul théorique est que le nombre d'agents pilotes varie dans le temps. En réalité, le nombre d'agents pilotes n'est pas constant. La figure 6-12 représente le nombre d'agents pilotes exécutés pendant l'expérimentation.

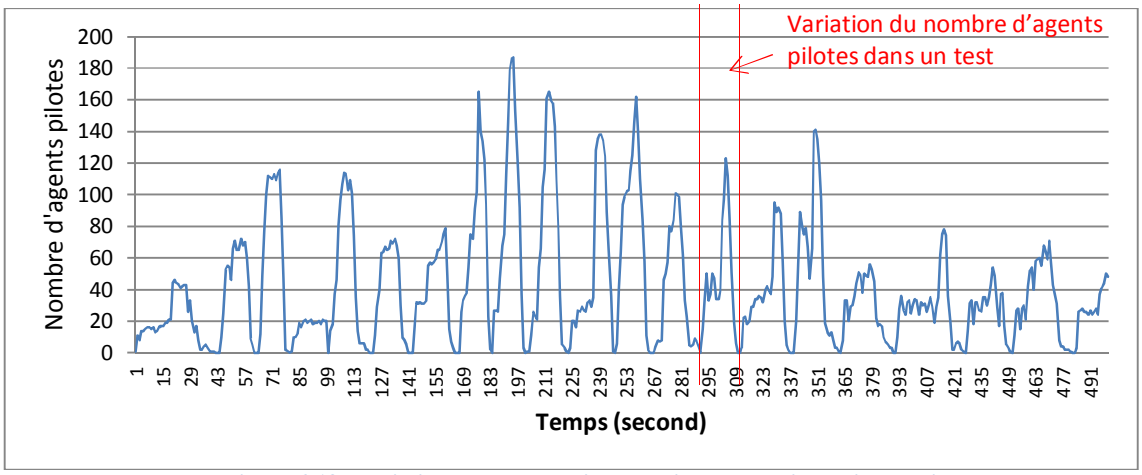


Figure 6-12: Variation du nombre d'agents pilotes dans l'expérimentation

Au début de chacun des 500 tests, le nombre d'agents est égal 0 parce qu'il n'y a pas de tâche d'utilisateur en attente. Lorsque les utilisateurs 1 et 2 soumettent leurs tâches, DIRAC

commence à soumettre des agents pilotes sur la grille. Le nombre d'agents augmente. Quand des agents pilotes sont soumis avec succès sur une machine, ils vont télécharger et exécuter les tâches des utilisateurs. Lorsqu'ils ont fini d'exécuter les tâches, ils s'arrêtent, leur nombre diminue et à la fin du test, ce nombre redevient nul.

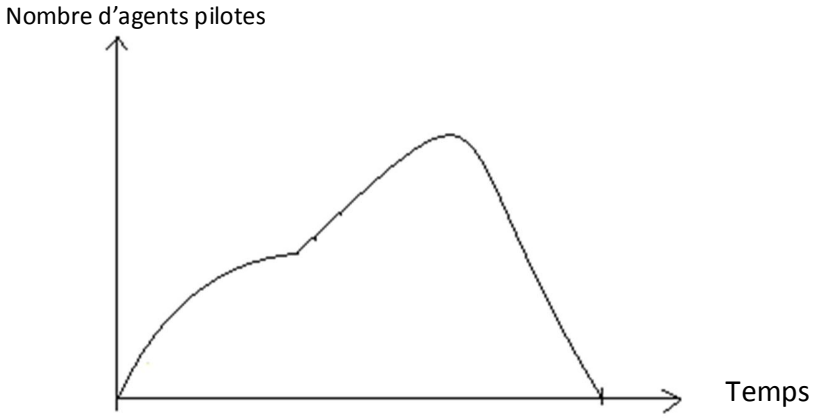


Figure 6-13: La variation du nombre d'agents pilotes dans un test

On peut voir dans un test que la variation du nombre d'agents pilotes ressemble à la figure 6-13. Au début du test, le nombre d'agent est égal 0 parce qu'il n'y a pas de tâches d'utilisateur. Ensuite, les utilisateurs 1 et 2 soumettent leurs tâches, DIRAC commence à soumettre des agents pilotes sur la grille. Le nombre d'agent augmente. Quand un agent pilote est soumis avec succès sur une machine, il va télécharger et exécuter les tâches des utilisateurs. Après que les agents pilotes téléchargent toutes les tâches des utilisateurs et terminent leur exécution, les agents pilotes vont se terminer. Le nombre d'agents pilotes va alors diminuer. A la fin du test, le nombre d'agents pilotes est égal à 0.

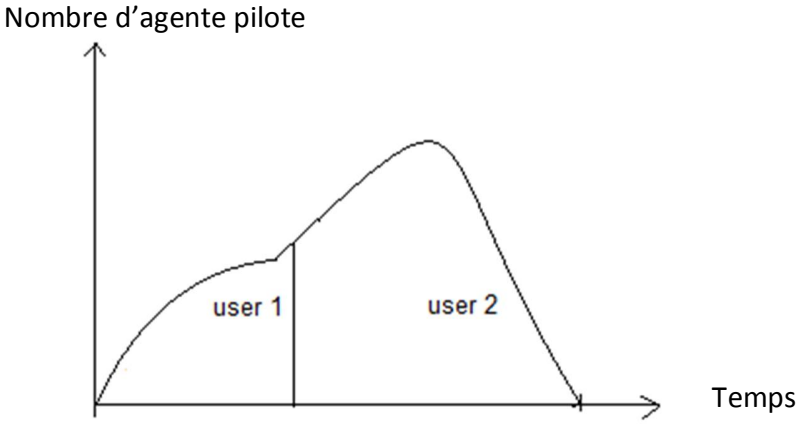


Figure 6-14: Utilisation du CPU dans la politique SPT

Dans le cas de la politique SPT, tous les agents pilotes exécutent en premier les tâches de l'utilisateur 1 parce que le nombre de tâches de l'utilisateur 1 est le plus petit. Après que toutes les tâches de l'utilisateur 1 se soient finies, les agents pilotes exécutent les tâches de l'utilisateur 2. La figure 6-14 représente l'utilisation des agents pilotes dans le cas de la politique SPT.

On note $M_{1\text{-}SPT}$ le nombre moyen d'agents pilotes qui s'exécutent pour l'utilisateur 1 et M_{SPT} le nombre moyen d'agents pilotes pendant le test. On a :

Le ralentissement de l'utilisateur 1 = $150/M_{1\text{-}SPT}$

Le ralentissement de l'utilisateur 2 = $350/M_{SPT}$

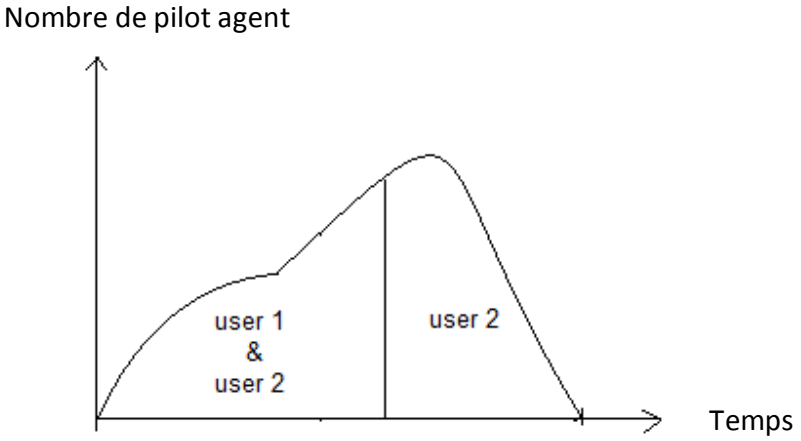


Figure 6-15: L'utilisation de CPU dans le cas de l'ordonnanceur de DIRAC

La figure 6-15 représente l'utilisation des agents pilotes dans le cas de l'ordonnanceur de DIRAC. Dans une première phase, les deux utilisateurs partagent des agents pilotes. Après

que toutes les tâches de l'utilisateur 1 se soient finies, l'utilisateur 2 prend tous les agents pilotes.

On dénote $M_{1\text{-}RR}$ le nombre moyen d'agents pilotes dans la première phase (exécution des tâches des utilisateurs 1 & 2), M_{RR} le nombre moyen d'agents dans ce test. On a :

$$\text{Le ralentissement de l'utilisateur 1} = 300/M_{1\text{-}RR}$$

$$\text{Le ralentissement de l'utilisateur 2} = 350/M_{RR}$$

Parce que le nombre d'agents pilotes augmente au début du test et que la durée de la première phase dans le cas de l'ordonnanceur de DIRAC est plus longue que dans le cas SPT, on a:

- ⇨ $M_{1\text{-}SPT} < M_{1\text{-}RR}$
- ⇨ $2*150/M_{1\text{-}SPT} > 300/M_{1\text{-}RR}$
- ⇨ 2*ralentissement de l'utilisateur 1 avec SPT > ralentissement de utilisateur 1 avec RR

En conclusion, nous observons expérimentalement sur la grille EGI que la politique d'ordonnancement SPT est meilleure que la politique RR, mais nous ne retrouvons pas la valeur théorique attendue pour le ralentissement expérimenté par les utilisateurs. Ce résultat s'explique notamment par le fait que la charge de la grille varie en fonction du temps et que les conditions dans lesquels les tests se déroulent ne peuvent pas être contrôlées.

En plus de cette première difficulté liée à la dynamique de la grille de production, nous n'avons pas été en mesure de comparer les performances des différentes politiques d'ordonnancement sur des scénarios impliquant un grand nombre d'utilisateurs pour un problème de certificat. En effet, les utilisateurs de DIRAC sur EGI doivent tous disposer d'un certificat personnel, garant de leur identité. Pour faire des tests, nous n'avons pas trouvé de méthode pour disposer d'un grand nombre de certificats avec lesquels déployer des tâches massives avec différentes politiques d'ordonnancement. Nous avons envisagé d'ouvrir le serveur DIRAC du LPC Clermont-Ferrand aux utilisateurs du logiciel AutoDock mais ce scénario s'est avéré incompatible avec la politique de sécurité du laboratoire. Il nous a donc été impossible de mesurer expérimentalement sur la grille EGI les performances des politiques SPT-SPT et SPT-RR.

Face à cette difficulté, nous nous sommes tournés vers une expérimentation avec la plate-forme HTCaaS sur PLSI en Corée du Sud.

6.2 Expérimentation avec HTCaaS sur PLSI

6.2.1 L'ordonnanceur initial de HTCaaS

HTCaaS (cf. section 2.4.4) maintient une file d'attente distincte d'agents pour chaque utilisateur. Chaque utilisateur soumet ses agents privés qui ne vont récupérer que ses propres tâches. HTCaaS utilise la politique Dynamic Fairness [41]. Il utilise une fonction d'attribution des ressources, RA (U), où U est un utilisateur du système.

$$RA(U) = \min\left(NumTasks(U), \frac{AvailableCores}{\sum_{p \in DU} Weight(p)} * Weight(U) \right) \quad (6.1)$$

Dans l'équation 6.1, NumTasks (U) est le nombre de tâches en attente dans la file d'attente de l'utilisateur U. AvailableCores est le nombre de ressources informatiques disponibles pour les agents (y compris les processeurs libres et ceux qui sont déjà attribués). Nous supposons une projection de un pour un entre un agent et un cœur de processeur dans le système. Un utilisateur est dit « exigeant » s'il a plus de tâches dans sa file d'attente que le nombre d'agents qui lui sont alloués en raison du manque de ressources de calcul disponibles. DU est l'ensemble des utilisateurs exigeants, N est le nombre d'utilisateurs exigeants. Weight(p) représente le poids d'un utilisateur p qui peut tenir compte de nombreux facteurs différents, tel que le nombre de tâches soumises par l'utilisateur p, le temps de calcul des tâches, sa priorité, etc. Dans l'implémentation actuelle de HTCaaS, cette fonction de poids est fixée pour tous les utilisateurs à 1 (Weight(p) = 1), de sorte que $\Sigma_{p\varepsilon DU}$Weight (p) est le nombre d'utilisateurs exigeants dans le système. La politique d'ordonnancement de HTCaaS ressemble à la politique Round Robin : le nombre d'agent réservé à chaque utilisateur est égal à Available_core/N.

Supposons par exemple qu'il y ait au total 100 CPUs libres à l'instant t_0 (figure 6.16). E_U dans la figure 6.16 est le nombre d'agents alloués à l'utilisateur U.

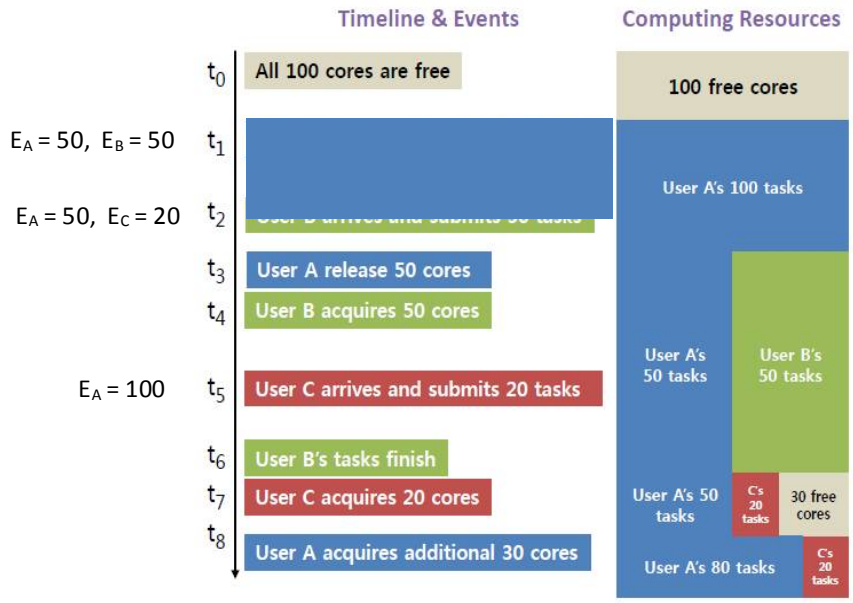


Figure 6-17: Gestion des tâches soumises avec la politique Dynamic Fairness

A l'instant t_1, l'utilisateur A arrive sur le système et soumet 400 tâches (le nombre de tâches est plus grand que le nombre total de ressources de calcul). Selon l'équation 7.1, RA (A) devient égal à 100 car il n'y a qu'un seul utilisateur exigeant dans le système (l'utilisateur exigeant est utilisateur qui a des tâches en attente, l'utilisateur satisfait est utilisateur dont toutes les tâches sont soumises au cluster). L'utilisateur A se voit allouer 100 agents.

A l'instant t_2, l'utilisateur B arrive au système et soumet 50 tâches. A ce moment, R (A) devient 50 et R (B) devient 50. Parce que R (A) a changé de 100 à 50 en raison de l'arrivée du nouvel utilisateur B, l'utilisateur A devrait libérer 50 agents pour assurer l'équité. Le système attend alors jusqu'à ce que les 100 agents finissent leurs tâches en cours. Ensuite, 50 agents continuent à exécuter des tâches de l'utilisateur A et 50 agents sont libérés. L'utilisateur B soumet 50 agents pour acquérir ces CPU libres et devient l'utilisateur satisfait quand toutes les tâches de B sont téléchargées par les agents. Le nombre d'utilisateurs exigeants diminue à 1 (seulement l'utilisateur A). Un utilisateur C arrive au système à l'instant t_5 et soumet 20 tâches. Le nombre d'utilisateurs exigeants devient deux (utilisateurs A et C) de sorte que R (A) devient 50 et R (C) devient 20. Par conséquent, jusqu'à ce que l'utilisateur C acquiert 20 agents (quand toutes les tâches de B sont finies), l'utilisateur A ne peut pas réclamer plus car 50 agents lui ont été affectés. Lorsque les tâches de l'utilisateur B sont terminées, l'utilisateur

C se voit allouer 20 agents et devient un utilisateur satisfait ; les 30 agents restants sont réalloués à l'utilisateur A.

6.2.2 Implémentation de SPT et SPT-SPT dans HTCaaS

Deux nouveaux ordonnanceurs SPT et SPT-SPT ont été intégrés dans HTCaaS. L'utilisateur peut changer facilement entre ces ordonnanceurs.

Nous avons mis en place l'ordonnanceur de la politique SPT dans HTCaaS avec une technique dite de « water-filling » (figure 6.17). Les utilisateurs sont triés par ordre croissant du nombre de jobs en attente. Les cœurs CPU sont alloués en premier à l'utilisateur ayant le moins de jobs en attente. S'il y a encore des cœurs libres, ils sont alloués à l'utilisateur suivant de la liste et ainsi de suite. L'algorithme 6.1 représente cette technique de « water filling » pour la politique SPT.

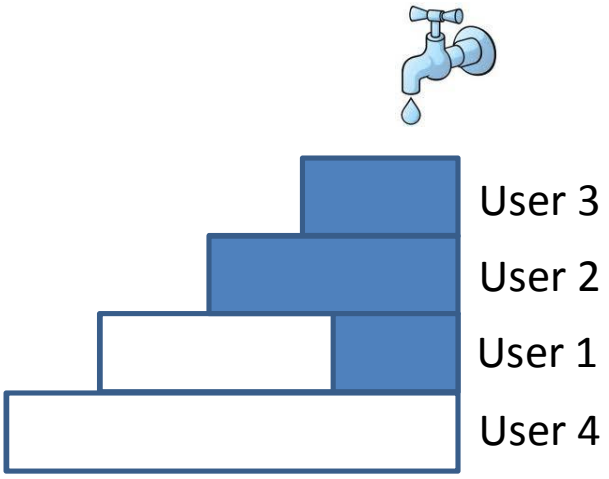


Figure 6-18: La technique dite de "Water Filling" pour la politique SPT

Algorithme 6.1 : Technique « water filling » pour la politique SPT

```
user_item contains [user_id, number_waiting_job]
List_user[] contains user_item
RA[i] is the number of resource allocated for user i

sort List_user[] by number of waiting jobs (ascending order)
N=number of waiting users in List_user[]
rest_CPU = number of cores CPUs in system
FOR (i = 1 to N)
DO
        IF (List_user[i].Number_waiting_job <= rest_CPU) THEN
                RA[List_user[i].user_id] = List_user[i].number_waiting_job;
                rest_CPU = rest_CPU – List_user[i].number_waiting_job;
        ELSE
                RA[List_user[i].user_id] = rest_CPU;
                rest_CPU = 0;
        END IF
END FOR
```

Comme nous le verrons au paragraphe 6.2.3.2, l'implémentation de la politique SPT par la technique de « water filling » réduit significativement le temps de calcul par rapport à la politique d'ordonnancement initiale de HTCaaS lorsqu'il y a un grand nombre d'utilisateurs.

Dans un premier temps, nous avons comparé les performances dans un scénario simple à deux utilisateurs : l'utilisateur 1 soumet 1000 tâches, l'utilisateur 2 soumet 500 tâches. Les tâches soumises par l'utilisateur dans ce test sont le même docking entre le ligand ZINC00001967 dans la base de donnée ZINC et la protéine 1EVE – la cible biologique de la maladie d'Alzeihmer. La durée de ce docking est 484.28 seconds sur le CPU Intel Duo Core 2 Ghz. L'utilisateur 2 arrive juste après l'utilisateur 1. Il y a 176 cœurs CPU dans le cluster.

La figure 6.18 représente la variation du nombre d'agents pilotes actifs pour chaque utilisateur en fonction du temps dans le cas de l'ordonnanceur SPT.

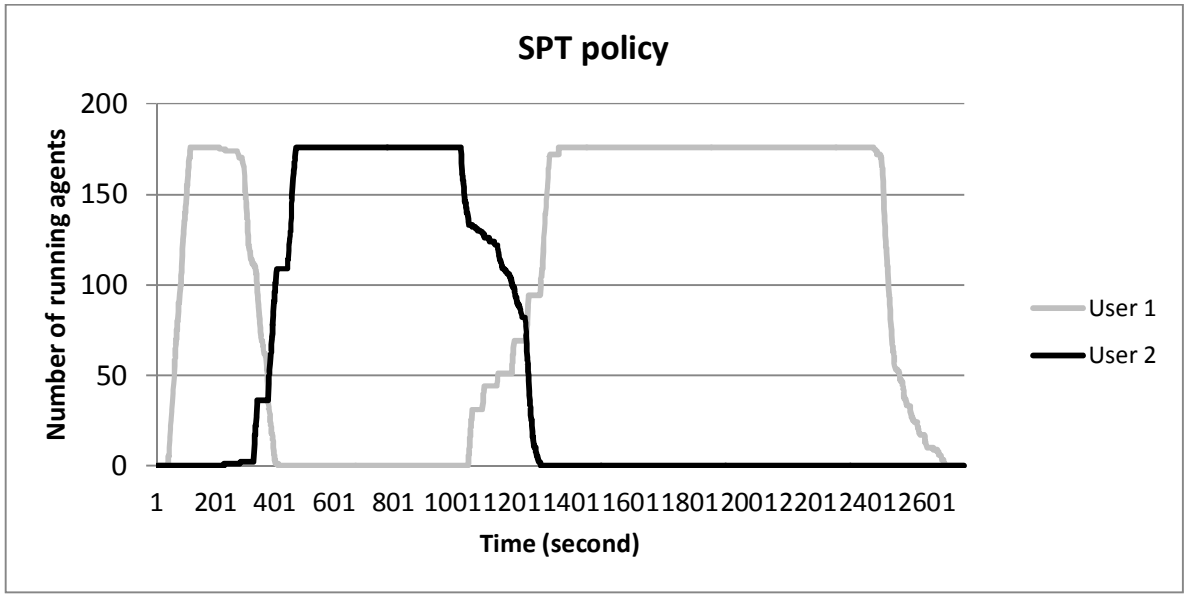


Figure 6-19:Utilisation du CPU dans le cas de l'ordonnanceur de la politique SPT

Tout d'abord l'utilisateur 1 occupe la totalité des 176 CPU. Lorsque l'utilisateur 2 arrive et que les agents pris par l'utilisateur 1 finissent l'exécution de leurs tâches, l'utilisateur 2 prend tous les 176 CPU. Une fois que les tâches de l'utilisateur 2 sont finies, l'utilisateur 1 reprend tous les 176 cœurs.

Pour implémenter l'ordonnanceur de la politique SPT-SPT, nous créons deux groupes d'utilisateurs « Normal » et « Data Challenge ». Dans chaque groupe, on implémente la politique SPT avec le technique « water filling ». Pour le groupe « Normal », on utilise le nombre de processeurs disponibles, égal à (p*nombre total de CPU). Pour le groupe « Data Challenge», le nombre de processeurs disponibles est égale à (1-p)*nombre total de CPU.

6.2.3 Expérimentation sur HTCaaS

6.2.3.1 SPT et l'ordonnanceur original de HTCaaS avec le scénario à 2 utilisateurs

a. Description du scénario

Dans ce scénario, il y a deux utilisateurs. L'utilisateur 1 soumet 1000 tâches. L'utilisateur 2 soumet 500 tâches. Les tâches soumises par l'utilisateur dans ce test sont le même docking entre le ligand ZINC00001967 dans la base de donnée ZINC et la protéine 1EVE – la cible

biologique de la maladie d'Alzeihmer. La durée de ce docking est 484.28 seconds sur le CPU Intel Duo Core 2 Ghz. L'utilisateur 2 arrive juste après l'utilisateur 1. Il y a 176 cœurs dans le cluster.

b. Résultat obtenu

L'expérience a été répétée 10 fois pour chaque politique (SPT et Dynamic Fairness). La durée de chaque test est d'environ 50 minutes. On calcule le ralentissement des utilisateurs dans chaque test. Le résultat est montré par la figure 6.19.

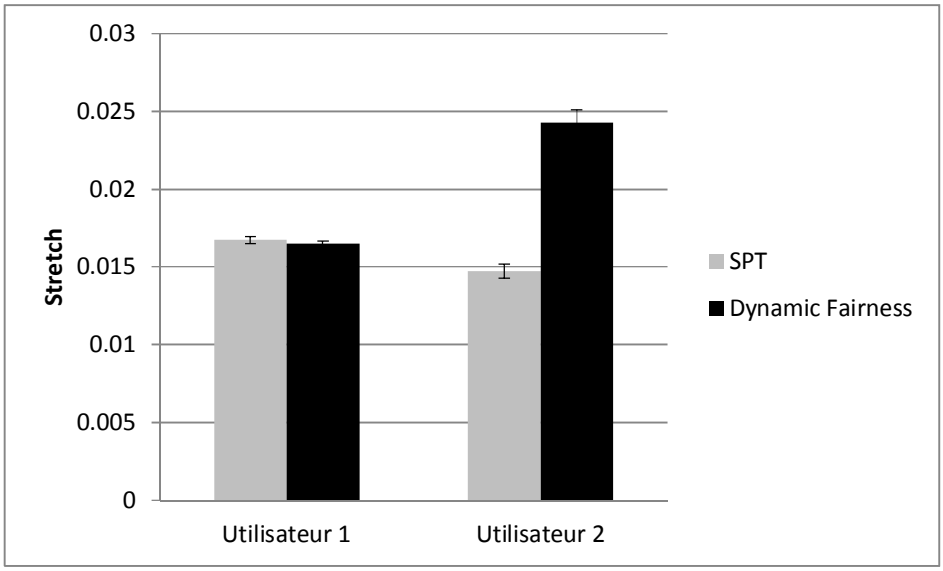


Figure 6-20: Ralentissements des utilisateurs

La barre d'erreur est petite parce que l'environnement du cluster ne change pas dans les tests différents, à la différence de la grille de production.

Comparé avec le calcul théorique du ralentissement pour les politiques SPT et Dynamic Fairness, on observe:

(1) au lieu d'être égal, le ralentissement de l'utilisateur 1 dans le cas SPT est légèrement supérieur à celui expérimenté dans le cas Dynamic Fairness.

(2) Le ralentissement de l'utilisateur 2 dans le cas SPT est supérieur à la moitié de son ralentissement dans le cas Dynamic Fairness.

Ces deux observations sont liées à la latence dans le transfert des ressources d'un utilisateur à l'autre.

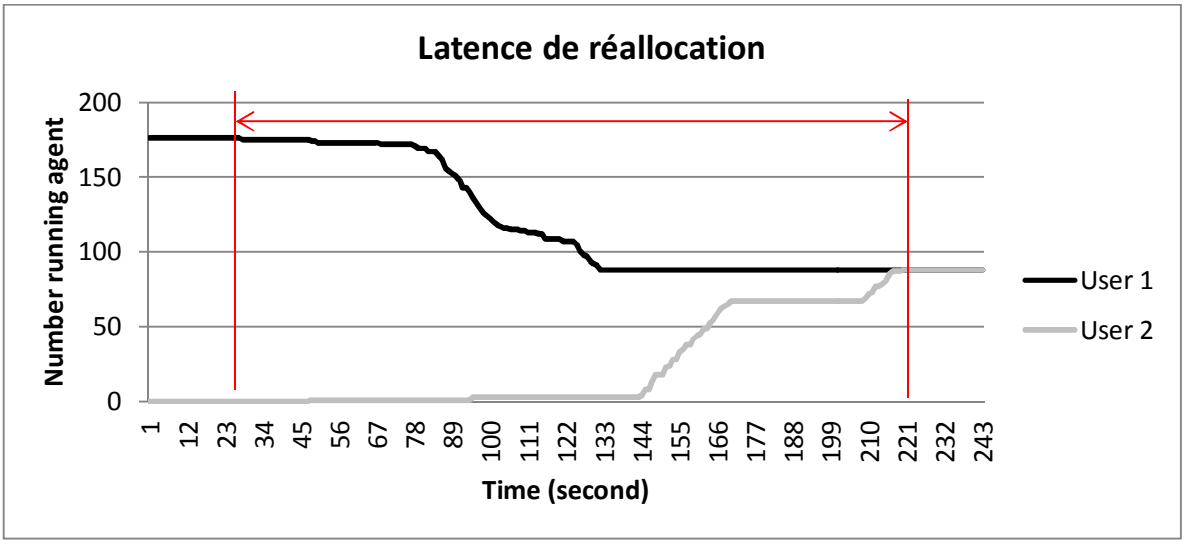


Figure 6-21: La latence de réallocation des ressources

La figure 6.20 représente la latence de réallocation dans le cas de la politique Dynamic Fairness. Tout d'abord, l'utilisateur 1 prend tous les 176 CPU. Lorsque l'utilisateur 2 arrive, les deux utilisateurs se partagent ces 176 CPU, c'est-à-dire que chaque utilisateur a 88 CPU. L'utilisateur 1 va libérer 88 CPU. Lorsque l'utilisateur 1 a fini de libérer ses ressources, l'utilisateur 2 soumet 88 agents pour récupérer ces 88 CPU libres. Dans cette période, on peut voir dans la figure que le nombre total d'agent est inférieur à 176, c'est-à-dire que quelques CPU sont « inoccupées » (idle).

La latence de réallocation des ressources est la somme de la latence de libération des ressources et de la latence de soumission des agents.

Dans le cas de la politique SPT, on doit changer 176 agents de l'utilisateur 1 à l'utilisateur 2, puis à nouveau 176 agents de l'utilisateur 2 à l'utilisateur 1. L'utilisateur 1 doit donc attendre le temps de réallocation de 2x176=352 agents avant que toutes ses tâches soient achevées.

Dans le cas de Dynamic Fairness, on doit changer 88 agents de l'utilisateur 1 à l'utilisateur 2 et 88 agents en sens inverse. L'utilisateur 1 doit attendre la latence de réallocation de 2x88 =176 agents. La latence de réallocation dans le cas de SPT est donc supérieure à celle observée dans le cas de Dynamic Fairness, ce qui explique pourquoi, sur la figure 6.19, le ralentissement de l'utilisateur 1 avec SPT est supérieur à celui expérimenté avec Dynamic Fairness.

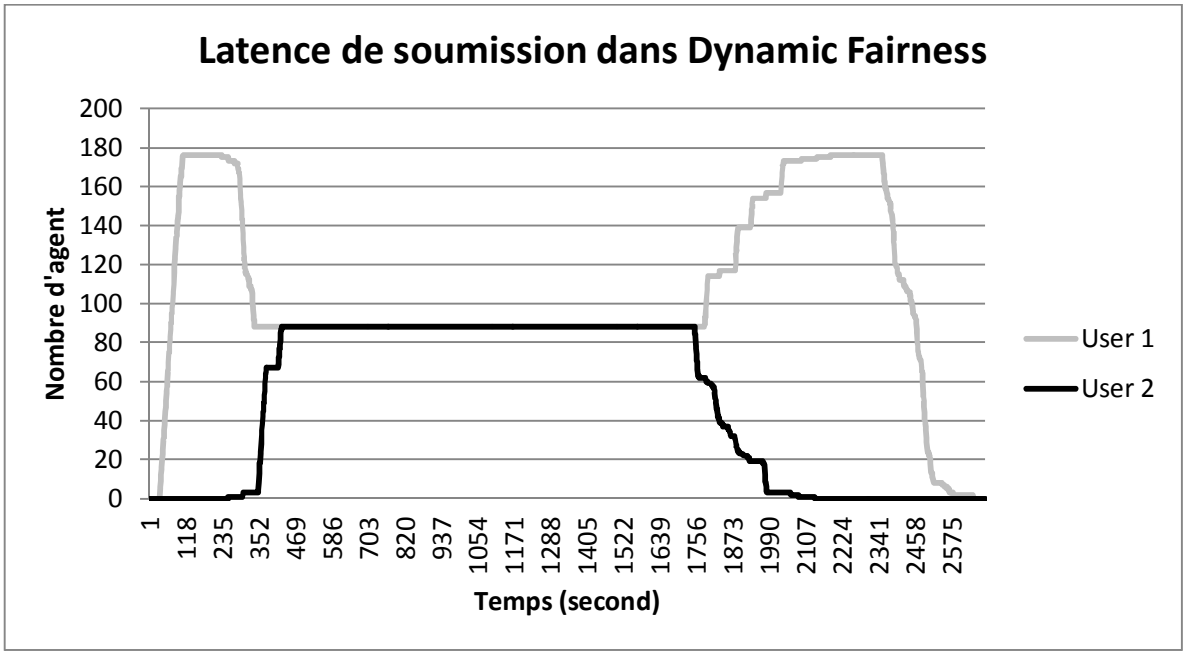


Figure 6-22: Latence de la soumission dans le cas de Dynamic Fairness.

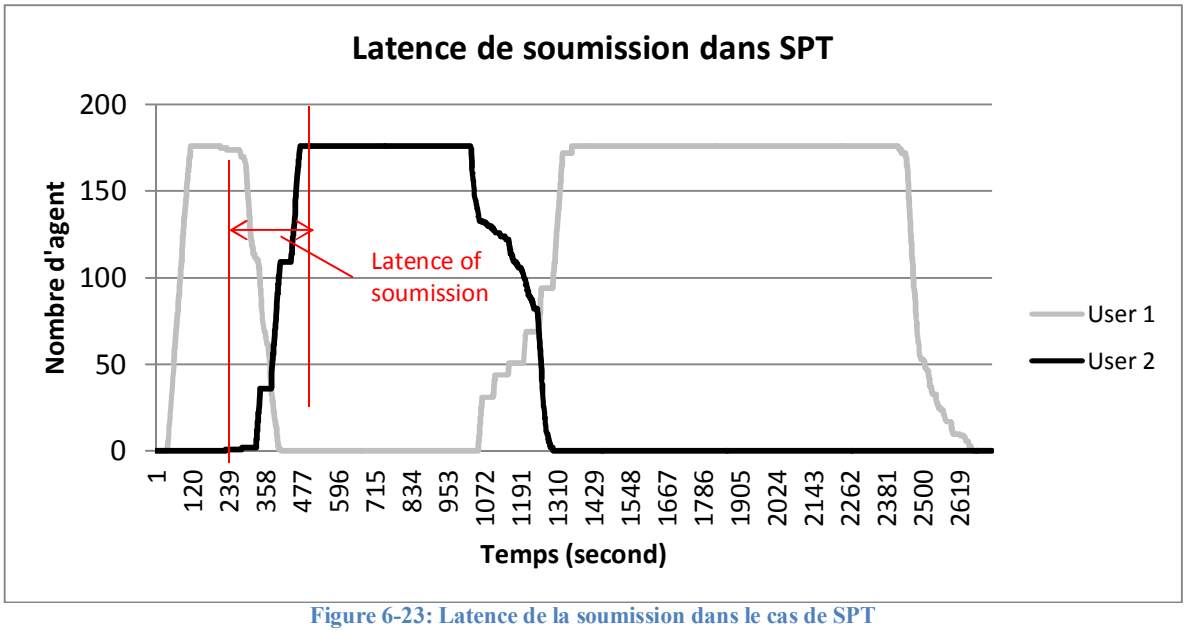


Figure 6-23: Latence de la soumission dans le cas de SPT

Les figures 6.21 et 6.22 représentent la latence de soumission dans le cas de Dynamic Fairness et SPT. L'utilisateur 2 doit soumettre 88 agents dans Dynamic Fairness et 176 agents dans SPT. Ainsi la latence de soumission est plus importante dans le cas de SPT. L'utilisateur 2 va donc expérimenter un ralentissement supérieur à la moitié de celui qu'il expérimente avec la politique Dynamic Fairness.

6.2.3.2 SPT, SPT-SPT et l'ordonnanceur original de HTCaaS avec le scénario de 1000 utilisateurs

a. Description du scénario

Dans ce scénario, nous utilisons un exemple de la charge d'utilisateur de criblage virtuel dans le jeu de données cas_00 (cf. section 5.1.3). Les 1000 premiers utilisateurs de cet exemple sont extraits avec :

- Le groupe « Normal » : 997 utilisateurs avec 30,125 tâches
- Le groupe « Data Challenge » : 3 utilisateurs avec 31,821 tâches

Les tâches soumises par l'utilisateur dans ce test sont le même docking entre le ligand ZINC00001967 dans la base de donnée ZINC et la protéine 1EVE – la cible biologique de la maladie d'Alzeihmer. La durée de ce docking est 484.28 seconds sur le CPU Intel Duo Core 2 Ghz. Le temps d'arrivée suit la distribution de Poisson. Le nombre de tâches soumises suit la distribution géométrique.

Nous avons testé ce scénario avec HTCaaS sur un cluster de 176 cœurs. Trois politiques d'ordonnancement ont été considérées : Dynamic Fairness, SPT et SPT-SPT, où le taux de ressource p alloué au groupe « Normal » est fixé à 0.7.

b. Résultat obtenu

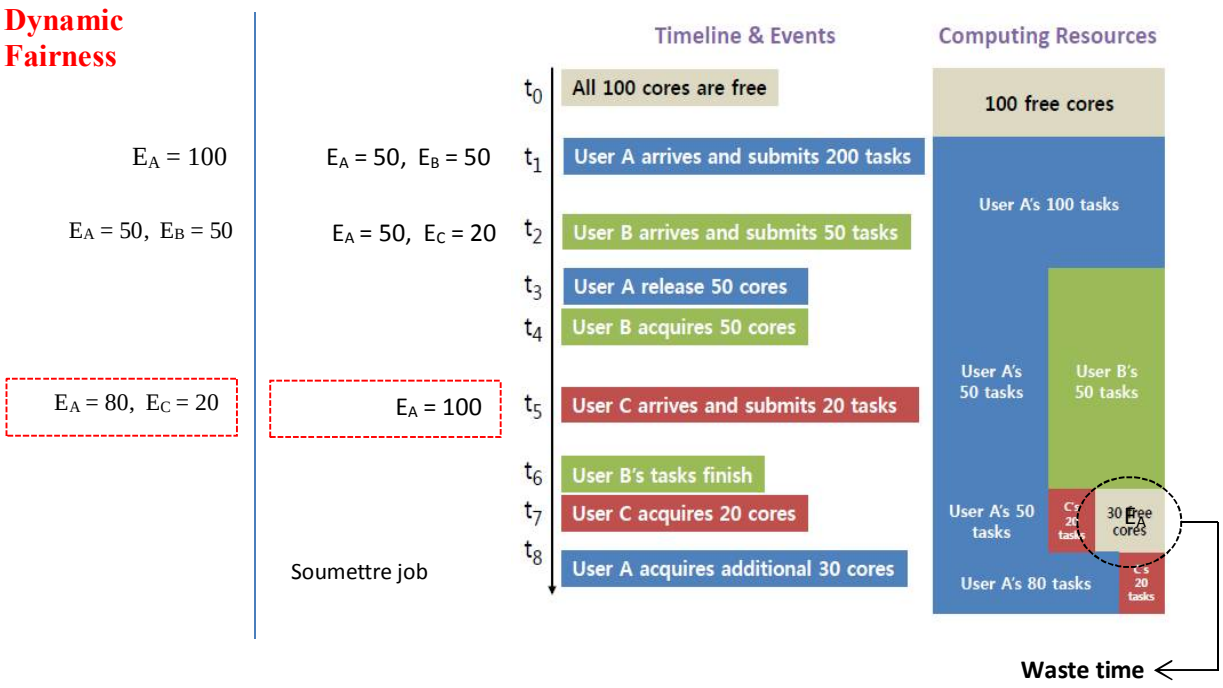
Les makespans des politiques sont représentés dans la table 7-3 :

Politique d'ordonnacement	Makespan
Dynamic Fairness	128292 s = 35h 38'
SPT	128528 s = 35h 42'
SPT-SPT (p=0.7)	123475 s = 34h 18'

Table 6-2: Makespan des différentes politiques d'ordonnancement

On peut voir dans la table 6.3 que le makespan est différent entre ces politiques. Les origines de cette différence sont la latence de soumission, la latence de réallocation (cf. section 6.2.3.1) et le temps de calcul perdu par la politique Dynamic Fainess.

Pour expliquer cette différence, considérons l'exemple des trois utilisateurs A, B et C présenté dans la section 6.2.1 pour un cluster disposant de 100 cœurs.



Technique « Water filling »

La figure 6.23 représente la différence entre la politique SPT (technique « water filling ») et la politique Dynamic Fairness dans cet exemple. EU est le nombre d'agents alloués à l'utilisateur U.

Avec la technique « Water filling », à l'instant t5, le nombre de cœurs est égal à 100, le nombre de jobs en attente de l'utilisateur 1 est de 250 (400 – 150 = 250) et celui de l'utilisateur 2 est de 20. On va remplir le besoin de tous les utilisateurs (cf. Algorithme 6.1). Alors le nombre d'agents de l'utilisateur 1 est égal à 80 et celui de l'utilisateur 3 est égal à 20. A l'instant t6, le système va soumettre 20 agents pour l'utilisateur C et 30 agents pour l'utilisateur A. La technique « water filling » va supprimer le temps perdu.

Dans cet exemple, le temps est perdu de l'instant t6 à l'instant t8. L'utilisateur A doit attendre l'utilisateur C qui envoie toutes les tâches aux agents pour soumettre les 30 nouveaux agents. Dans cette période, les 30 CPUs sont libres pendant que l'utilisateur A est toujours en attente.

Les différences observées entre les trois politiques sur le makespan peuvent se résumer ainsi :

- comparaison entre SPT et Dynamic Fairness : bien que la latence de réallocation de ressource dans la politique SPT soit supérieure à celle liée à la politique Dynamique Fairness (cf. section 6.2.3.1), la technique de « water filling » permet de calculer mieux le nombre d'agents pour chaque utilisateur, et le temps perdu diminue.
- comparaison entre SPT-SPT et Dynamic Fairness: la politique SPT-SPT réserve un nombre fixe d'agents pour le groupe « Data Challenge », il réduit donc le nombre d'agents échangés entre les utilisateurs. Par conséquent, la latence de réallocation diminue. De plus la technique "Water Filling" calcule mieux le nombre d'agents réservés pour chaque utilisateur et le temps perdu diminue aussi.
- comparaison entre SPT-SPT et SPT : La politique SPT-SPT réserve un nombre fixe d'agents pour le groupe « Data Challenge », il réduit donc le nombre d'agents échangés entre les utilisateurs. Par conséquent, la latence de réallocation diminue.

	Groupe « Normal »	Groupe « Data Challenge »
Dynamic Fairness	27.23	0.143
SPT	2.02	0.125
SPT-SPT (p=0.7)	3.62	0.035

Table 6-3: Ralentissement maximal des groupes

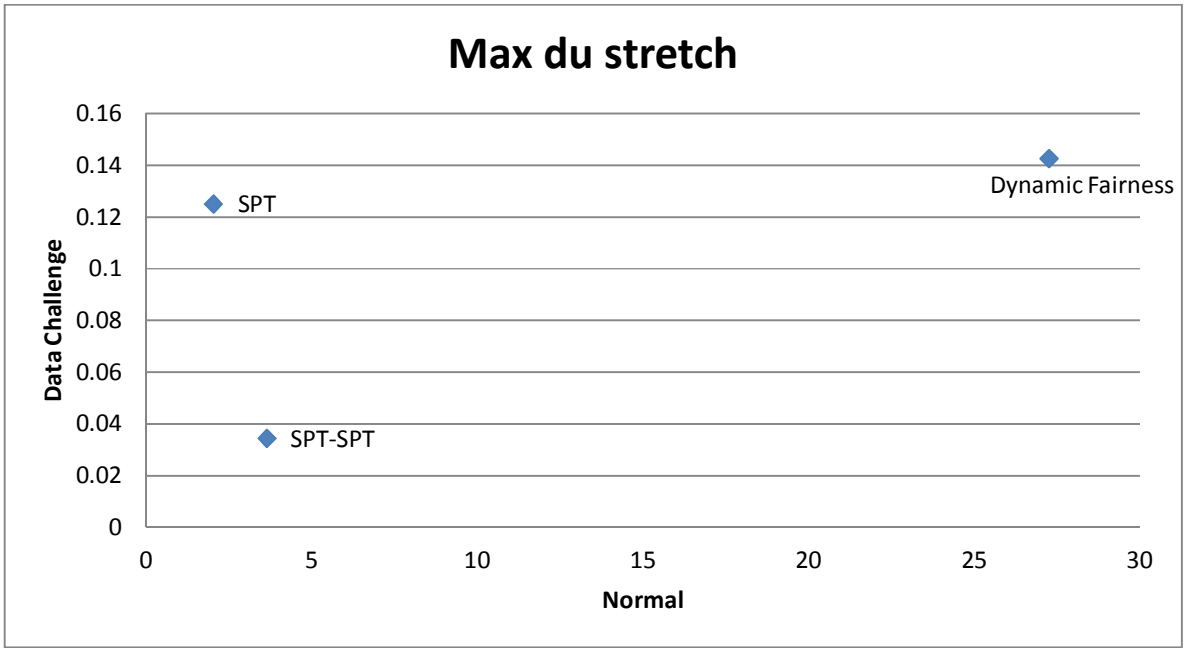


Figure 6-24: Ralentissement maximal des groupes d'utilisateurs normaux et « data challenge » pour les trois politiques testées

La table 6.3 et la figure 6.24 représentent le ralentissement maximal S_{max} des groupes d'utilisateurs normaux et « Data Challenge » pour les trois politiques testées (Dynamic Fairness, SPT et SPT-SPT).

Le S_{max} du groupe "Normal" dans la politique Dynamic Fairness est bien supérieur au Smax dans la politique SPT. Dans ce groupe « Normal », le S_{max} augmente d'un facteur de 1.79 entre les deux politiques SPT-SPT et SPT.

Pour le groupe « data challenge » nous constatons que la politique Dynamic Fairness est à nouveau moins performante que la politique SPT. La politique SPT-SPT réduit d'un facteur 4 le ralentissement maximal par rapport à la politique SPT.

S_{moyen}	Dynamic Fairness	SPT	SPT-SPT (p=0.7)
Normal	5.809	0.622	0.703
Data Challenge	0.0751	0.068	0.032

Table 6-4: Ralentissement moyen des groupes

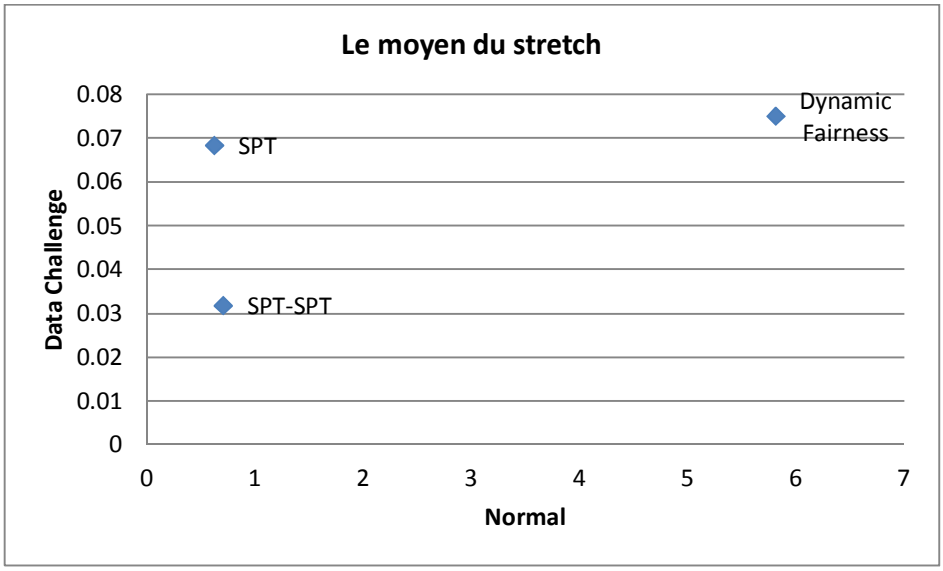


Figure 6-25: Ralentissement moyen des groupes d'utilisateurs normaux et « data challenge » pour les trois politiques testées

La figure ci-dessus montre que pour le groupe « Normal », le Smoyen augmente d'un facteur de 1.13 entre les deux politiques SPT-SPT et SPT. Le Smoyen du groupe "Normal" dans la politique Dynamic Fairness est bien supérieur au Smoyen dans la politique SPT. Pour le groupe « Data Challenge » nous constatons que la politique Dynamic Fairness est moins performante que la politique SPT. La politique SPT-SPT réduit d'un facteur 2.125 le ralentissement moyen par rapport à la politique SPT.

En résumé, la politique SPT-SPT est meilleure que SPT et Dynamic Fairness dans les tests de scénario à deux utilisateurs et de scénario à 1000 utilisateurs. Elle diminue significativement le ralentissement des utilisateurs « data challenge » tout en augmentant un peu le celui des utilisateurs dans le groupe « Normal ». De plus, elle diminue le makespan sur le système HTCaaS.

6.2.4 Validation des résultats expérimentaux

Les résultats expérimentaux obtenus avec la plate-forme HTCaaS sur PLSI ne peuvent être directement comparés à ceux obtenus avec SimGrid et décrits dans le chapitre précédent, parce que nous avons considéré dans la simulation que les agents pilotes étaient partagés par les utilisateurs alors qu'ils sont privés dans la version actuelle de HTCaaS. Pour applier à

cette situation, nous avons modifié le simulateur pour simuler l'ordonnancement de HTCaaS. Pour cela, nous introduisons des agents privés dans le simulateur de HTCaaS : chaque utilisateur a sa propre file d'attente des tâches. L'agent d'un utilisateur télécharge seulement ses tâches pour les exécuter.

Le simulateur tourne dans la configuration du cluster du KISTI à 176 cœurs. La vitesse des cœurs est mesurée sur le temps de calcul d'une tâche de docking soumise aux 176 CPUs du cluster (figure 6.26). Ce résultat permet de calculer la vitesse relative entre les CPU du cluster.

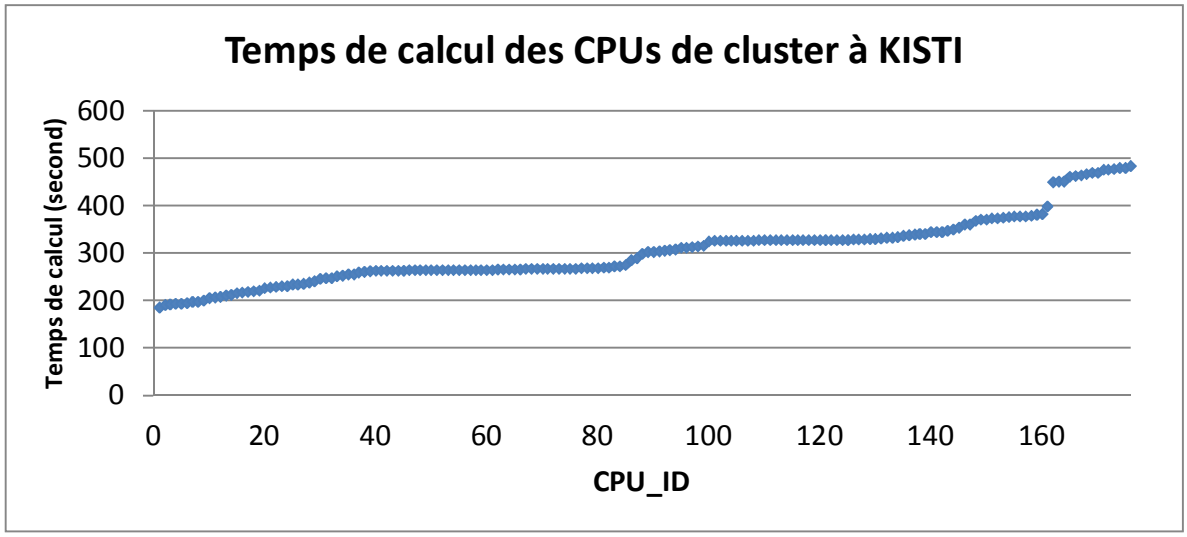


Figure 6-26: Distribution des temps de calcul d'une même tache de docking sur les 176 cœurs du cluster du KISTI

Nous avons refait les scénarios expérimentaux sur le simulateur de HTCaaS avec la même configuration de l'infrastructure de cluster à KISTI et comparer les résultats.

6.2.4.1 Scénario à 2 utilisateurs

Nous avons testé 10 fois le scénario à deux utilisateurs (cf. section 6.2.3.1). Les figures 6.26 et 6.27 montrent la variation simulée et la variation expérimentale du nombre d'agents de chaque utilisateur, mettant en évidence l'excellent accord entre le simulateur et l'expérimentation. La seule différence entre les deux résultats est due à la vitesse différente entre les machines du cluster. En effet, à chaque test, une tâche va s'exécuter aléatoirement sur

des machines de vitesses différentes, ce qui induit une différence entre les résultats expérimentaux et les simulations.

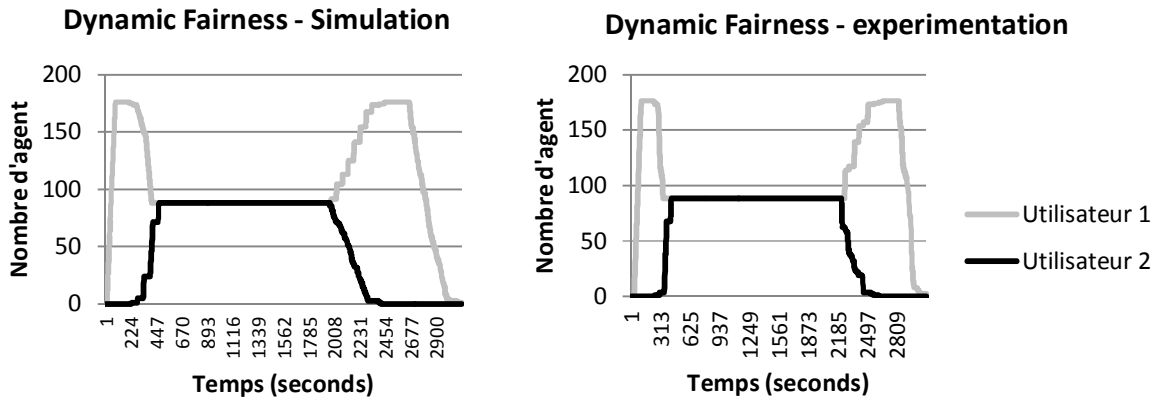


Figure 6-27: Variation du nombre d'agents de chaque utilisateur dans la politique Dynamic Fairness

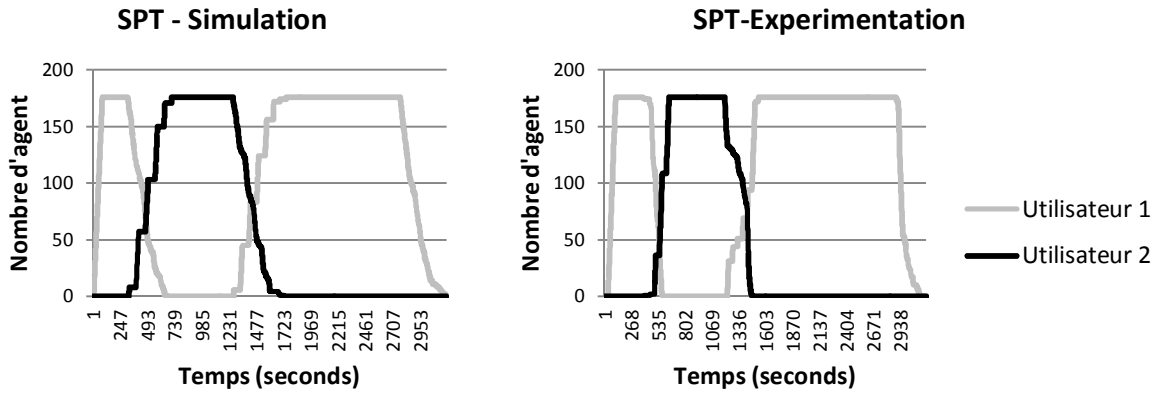


Figure 6-28: Variation du nombre d'agents dans la politique SPT

La figure 6.29 montre le ralentissement maximal des utilisateurs à chacun des 10 tests. On peut voir que les nuages des résultats de simulation et de l'expérimentation sont proches dans les deux politiques SPT et Dynamic Fairness.

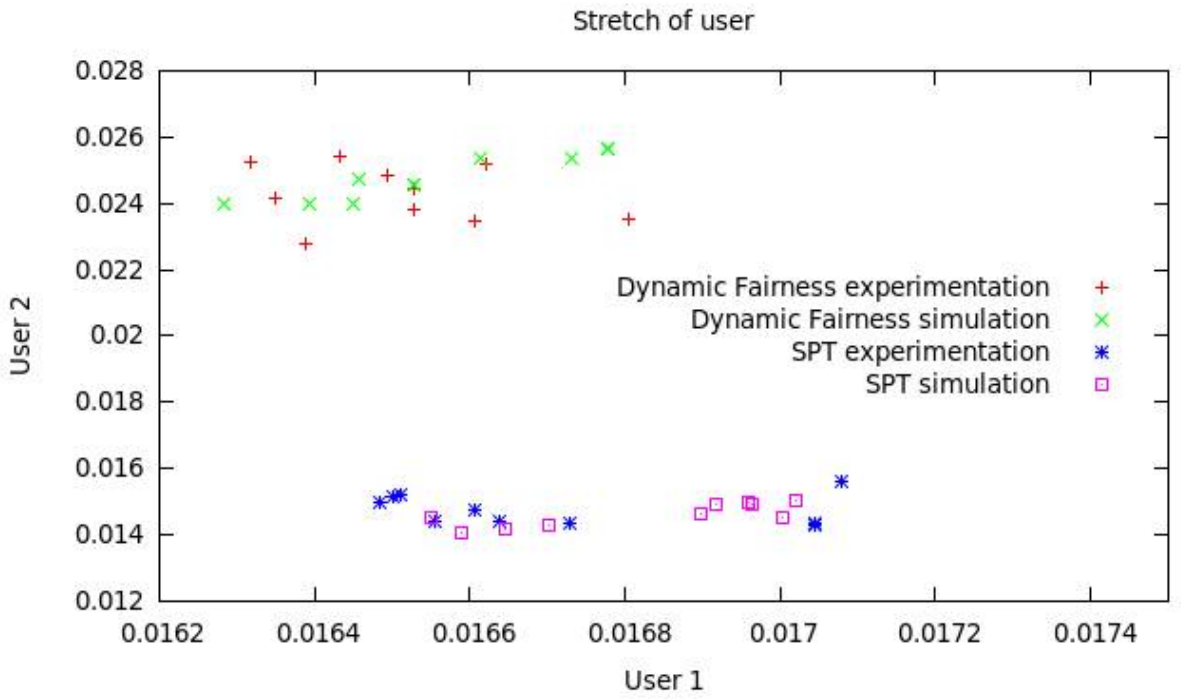


Figure 6-29: Ralentissement maximal des utilisateurs

SPT	Simulation		Expérimentation	
	User 1	User 2	User 1	User 2
Moyen	0.0168	0.0146	0.0167	0.0147
Ecart-type	0.0002	0.0003	0.0002	0.0004

Table 6-5: Moyenne et écart-type du ralentissement des utilisateurs avec la politique SPT

Dynanic Fairness	Simulation		Expérimentation	
	User 1	User 2	User 1	User 2
Moyen	0.0166	0.0248	0.0165	0.0243
Ecart-type	0.0002	0.0007	0.0001	0.0008

Table 6-6: Moyenne et écart-type du ralentissement des utilisateurs avec la politique SPT

Les tables 6.5 et 6.6 donnent la moyenne et l'écart-type du ralentissement des utilisateurs obtenues par la simulation et l'expérimentation respectivement pour les politiques Dynamic Fairness et SPT. La comparaison des résultats montre un excellent accord.

6.2.4.2 Scénario avec 1000 utilisateurs

Dans cette section, nous reprenons le scénario avec 1000 utilisateurs (cf. section 6.2.3.2). Nous avons testé la méthode Dynamic Fairness, SPT et SPT-SPT pour différentes valeurs du paramètre p définissant la fraction de ressources allouée au groupe « normal » : 0.1, 0.3, 0.5, 0.7, 0.9 et 1. Chaque méthode a été exécutée 50 fois. Les résultats sont représentés sur la figure 6.30.

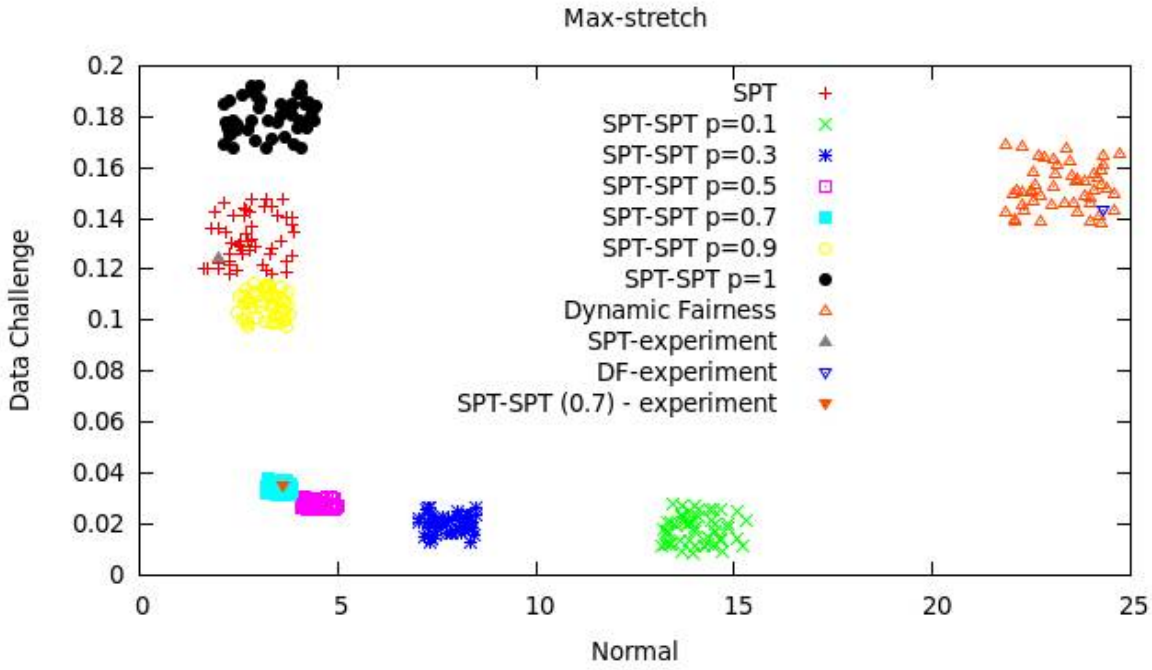


Figure 6-30 a: Ralentissement maximal des utilisateurs

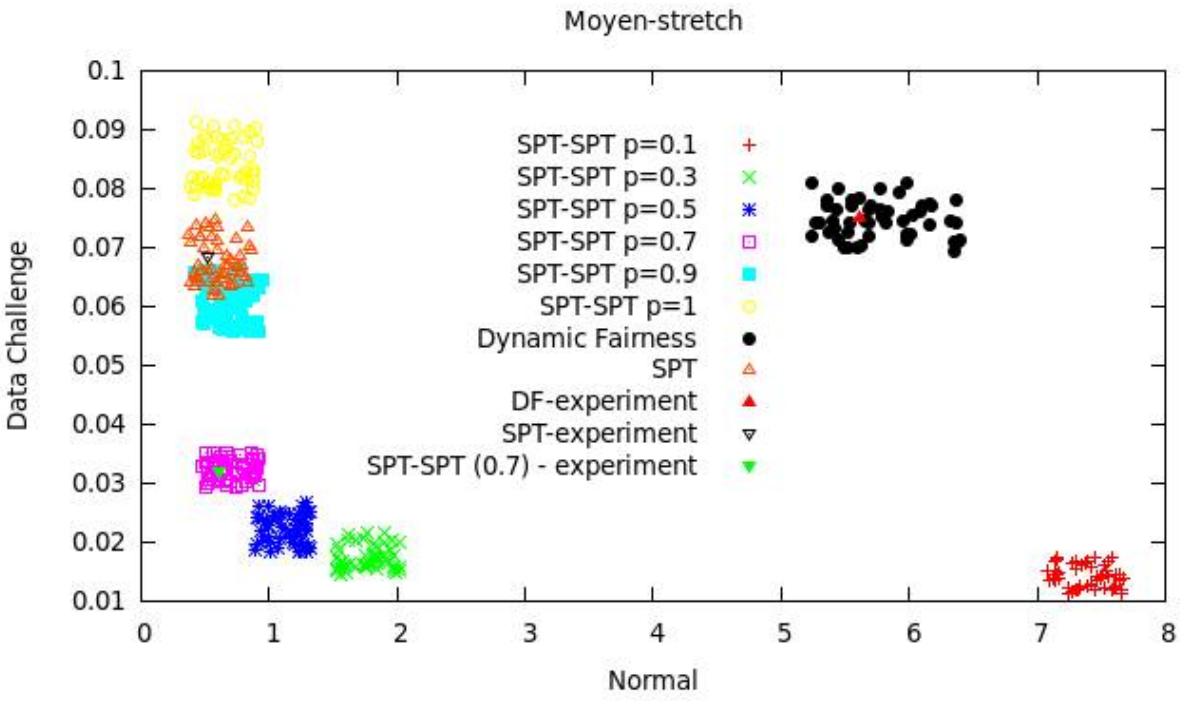
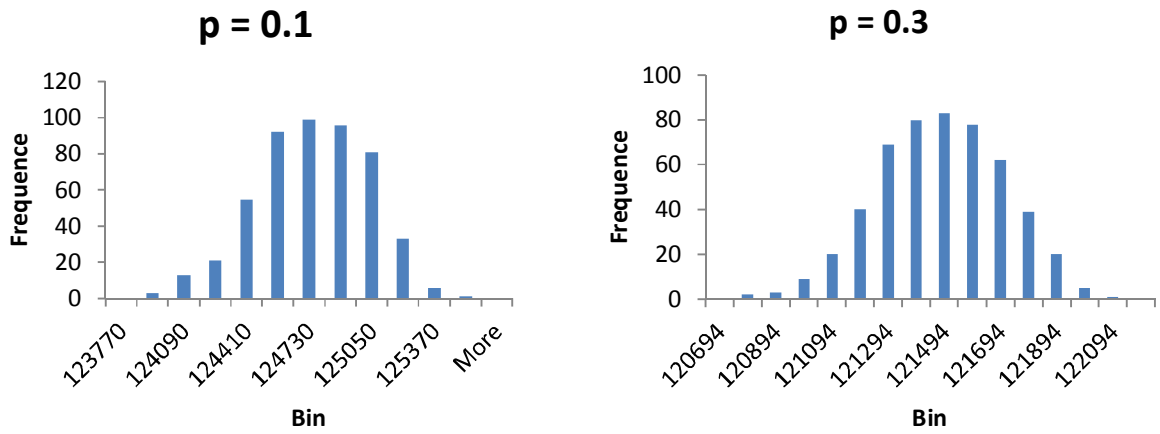


Figure 6-31 b: Ralentissement moyen des utilisateurs

En outre, nous avons étudié la variation du makespan selon le changement du paramètre p dans la politique SPT-SPT. Nous avons testé sur le scénario de 1000 utilisateurs la politique SPT-SPT pour p = 0, 0.1, 0.2, … 0.9, 1. Chaque valeur de p est testée 500 fois. La figure 6.31 représente l'histogramme des makespans obtenus selon les différentes valeurs de p.



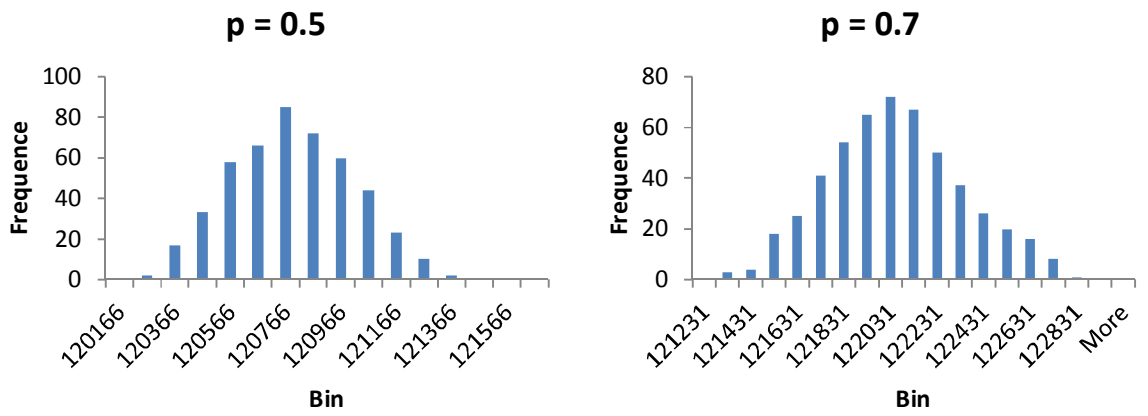


Figure 6-32: Histogramme des valeurs du Makespan pour 500 répétitions du scénario à 1000 utilisateurs

Nous pouvons voir que le makespan suit une distribution normale. Nous avons pris la moyenne des résultats pour comparer le makespan pour différentes valeurs du paramètre p (figure 6-32).

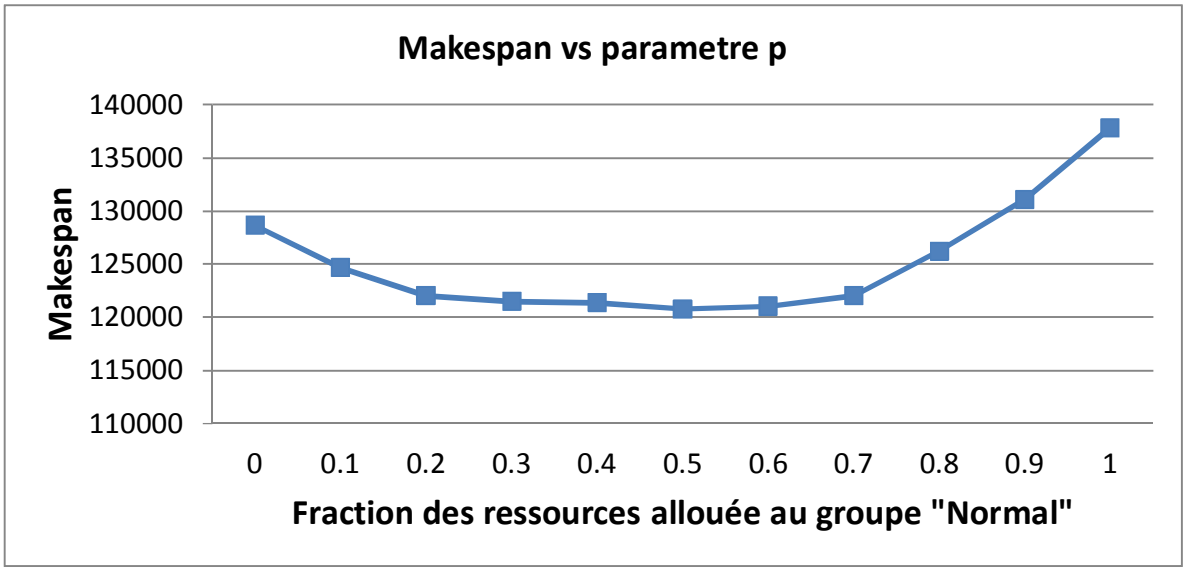


Figure 6-33: Variation de makespan contre le paramètre p

Le minimum du makespan est obtenu p = 0.5, ce qui est cohérent avec le fait que le nombre total de tâches soumises par les utilisateurs du groupe « Normal » est environ égal au nombre de tâches soumises par les utilisateurs du groupe « Data Challenge » dans le scénario étudié. Ainsi lorsque p = 0.5, le nombre de réallocations des ressources entre les utilisateurs est minimisé, ce qui réduit la latence de réallocation.

Pour p <0.5, le makespan est plus petit que pour p > 0.5. En effet, quand p < 0.5, les tâches des utilisateurs du groupe « Normal » ont tendance à démarrer plus tard. Comme

l'ordonnanceur a plus d'informations sur le nombre de tâches des utilisateurs du groupe « Normal », la file d'attente est bien triée. Lorsque $p > 0.5$, les utilisateurs du groupe « Normal » ont tendance à démarrer plus tôt. L'ordonnanceur a moins d'informations sur le nombre de tâches des utilisateurs. Il est probable qu'un utilisateur arrive avec moins de tâches que l'utilisateur en cours d'exécution. La ressource doit alors être réallouée au nouvel utilisateur et il y a de la latence de réallocation.

6.3 Conclusion

Nous avons présenté dans ce chapitre les résultats d'expérimentation de plusieurs politiques d'ordonnancement de deux plates-formes à agents pilotes, DIRAC et HTCaaS, sur deux environnements de production, la grille européenne EGI et un cluster du KISTI.

Sur la grille européenne EGI, nous avons utilisé un serveur DIRAC dédié installé au LPC Clermont-Ferrand pour tester le scénario à deux utilisateurs sur l'Organisation Virtuelle Biomed. Le test expérimental a confirmé que la politique SPT était plus performante que la politique RR. Par contre, nous n'avons pas pu étudier les scénarios multiutilisateurs simulés avec SimGrid parce que nous nous sommes heurtés à la difficulté de disposer d'un nombre important de certificats. Pour pallier à cette difficulté, il est possible d'étudier les traces d'une plate-forme offrant un service à une large communauté d'utilisateurs comme VIP ou FG-DIRAC en jouant sur la politique d'ordonnancement mais cela peut occasionner une gêne ou une dégradation du service offert aux utilisateurs. Qui plus est, la charge de la grille affecte significativement le nombre d'agents soumis, les temps de réponse, … Seule une grille dédiée à la recherche comme Grid5000 peut permettre une véritable comparaison avec la théorie et la simulation.

Sur le cluster du KISTI, nous avons testé les scénarios à 2 utilisateurs et à 1000 utilisateurs avec la plate-forme à agents pilotes HTCaaS. Celle-ci utilisant des agents privés, nous avons modifié notre simulation dans SimGrid pour introduire cette spécificité, aboutissant à un accord très satisfaisant entre les ralentissements simulés et observés expérimentalement.

Il en ressort que la politique SPT-SPT, proposée au chapitre 5, améliore significativement l'expérience des très gros utilisateurs de ressources (« data challenge » users) en ne dégradant que faiblement celle des utilisateurs normaux. C'est donc une piste sérieuse à envisager si une plate-forme à agents pilotes doit être partagée par tout un spectre d'utilisateurs aux besoins variés sur une grille ou une fédération de clouds académiques.

CHAPITRE 7 : CONCLUSION ET PERSPECTIVE

7.1 Conclusion

La découverte de médicaments *in silico* est l'une des stratégies les plus prometteuses visant à accélérer le processus de développement de médicaments. Elle permet aux chercheurs de trier (screen) rapidement les grandes bases de données de médicaments potentiels qui nécessiteraient autrement un travail fastidieux et de longue durée en laboratoire selon les méthodes traditionnelles de découverte de médicaments. L'Institut de Chimie des Produits Naturels de l'Académie des Sciences du Vietnam (INPC) collecte des échantillons issus de la biodiversité locale et détermine la structure tridimensionnelle des molécules isolées. Les chercheurs de l'INPC ont besoin de lancer de façon massive et répétitive des tâches de docking pour sélectionner les cibles biologiques potentiellement inhibées par les ligands isolés à partir de la biodiversité au Vietnam.

L'utilisation d'une plate-forme permet de faciliter le déploiement des calculs à grande échelle et de masquer les difficultés techniques liées à la grille et le cloud. De plus, il y a aussi la difficulté pour l'INPC de gérer sa propre plate-forme dédiée et la nécessité de partager cette plate-forme avec d'autres utilisateurs. Il faut trouver une politique d'ordonnancement pour partager correctement les ressources de la grille ou le cloud entre les utilisateurs.

Le travail décrit dans ce manuscrit s'est concentré sur la recherche d'ordonnancement d'une plate-forme de criblage virtuel sur la grille et le cloud. Nous avons étudié l'ordonnancement des jobs des utilisateurs dans la file d'attente et évalué la pertinence et l'impact des différentes politiques d'ordonnancement (FIFO, SPT, LPT, RR, FairShare, SPT-SPT et SPT-RR) sur l'expérience de l'utilisateur. Le critère optimal utilisé dans notre recherche est le stretch, une mesure de l'expérience de l'utilisateur sur la plateforme.

Dans une première étape, nous avons simulé le fonctionnement des applications de criblage virtuel sur la plate-forme pilote agent afin de comparer les politiques d'ordonnancement. Selon le résultat de simulation, la politique SPT a montré le meilleur résultat par rapport aux

politiques FIFO, LPT, RR et FairShaire pour optimiser le stretch des utilisateurs criblage virtuel. De plus, la recherche de la charge de la grille de calcul a montré qu'il existe deux types d'utilisateurs: de nombreux utilisateurs qui soumettent un petit nombre de tâches (utilisateur normal) et un petit nombre d'utilisateurs qui soumettent un grand nombre de tâches (utilisateur Data Challenge). La politique SPT n'est pas bonne pour les utilisateurs Data Challenge parce qu'ils doivent attendre une longue série d'utilisateurs normaux. Nous avons proposé des nouvelles politiques d'ordonnancement nommées SPT-SPT et SPT-RR. Ces politiques utilisent deux files d'attente différentes : une file d'attente pour l'utilisateur normal et une autre pour l'utilisateur Data Challenge. Ensuite on définit la probabilité d'envoyer une tâche de chaque file d'attente à l'agent pilote lorsque l'agent pilote demande les tâches aux utilisateurs. Les résultats des simulations ont montré que la politique SPT-SPT a un meilleur résultat sur le stretch de l'utilisateur que la politique SPT originale. Le stretch du groupe d'utilisateurs de Data Challenge diminue et le stretch des utilisateurs normaux est presque inchangé.

Dans une deuxième étape, nous avons fait des expérimentations de politiques sur l'environnement réel. La politique SPT a été mise en œuvre sur une instance de la plate-forme DIRAC au LPC – Clermont Ferrand et testé sur l'infrastructure de la grille EGI via l'organisation virtuelle Biomed. Les résultats expérimentaux sont en bon accord avec la simulation et confirment que l'algorithme SPT améliore considérablement l'expérience d'utilisateur. Nous avons aussi implémenté les politiques SPT et SPT-SPT sur la plate-forme HTCaaS au KISTI, en Corée. Les tests ont été effectués sur l'infrastructure du cluster au KISTI. On utilise des scénarii avec un grand nombre d'utilisateurs dont des utilisateurs normaux et des utilisateurs Data Challenge. Le résultat obtenu a montré que SPT est meilleure que la politique Dynamic Fairness (la politique originale de HTCaaS) pour optimiser le stretch d'utilisateur. De plus, on trouve aussi que la politique SPT-SPT a de meilleurs résultats que la politique SPT. Elle a augmenté l'expérience d'utilisateur Data Challenge et il n'a pas eu beaucoup d'impact sur l'expérience d'utilisateurs normaux.

Nous avons développé aussi un portail web pour le criblage virtuel, qui permet aux utilisateurs (biologistes, chimistes, bio-informaticiens...) à l'INPC d'envoyer des jobs de docking sur la grille de calcul et de récupérer les résultats à travers ce portail, pour accélérer leur recherche de nouveaux médicaments. Ce travail est décrit en annexe du manuscrit car il a été fait par un étudiant en stage à l'Institut de la Francophonie pour l'Informatique à Hanoï.

Nous offrons aux utilisateurs finaux, qui ne sont pas des experts ni en informatique ni en technologie de la grille, une interface conviviale et facile à utiliser sans qu'ils se soucient de la complexité du portail. Ce portail utilise la plate-forme DIRAC pour soumettre des jobs d'utilisateur sur la grille EGI via l'organisation virtuelle Biomed.

Cependant, il reste encore du travail sur l'évaluation de l'algorithme SPT-SPT sur l'environnement réel de la grille et du cloud. Nous avons effectué l'expérimentation de l'algorithme SPT-SPT sur le cluster du KISTI via la plate-forme HTCaaS avec un grand nombre d'utilisateurs virtuels créés par l'administrateur de HTCaaS. Mais sur la grille et le cloud, la simulation avec des utilisateurs virtuels est impossible, il faut des certificats de véritables utilisateurs pour soumettre des jobs sur la grille et le cloud. Il est difficile au niveau administration de créer un grand nombre de certificats pour ce test. Dans l'avenir, nous pensons installer SPT-SPT sur la plate-forme DIRAC et évaluer la performance de cette politique sur la charge d'utilisateur réel sur une longue période. Pour cause de contrainte de temps limité, l'expérimentation de la politique SPT-SPT sur l'environnement de la grille et le cloud n'a pas été faite dans le cadre de ma thèse.

7.2 Perspective

7.2.1 Perspective de la politique de l'ordonnancement SPT-SPT

Actuellement, le paramètre p de la politique SPT-SPT est configuré par l'administrateur de la plate-forme. A partir de la charge des groupes Normal et Data Challenge, il va falloir changer le paramètre p pour avoir la meilleure performance. Dans le futur, on peut améliorer la politique pour que le paramètre puisse changer automatiquement pour s'adapter à l'environnement. La politique va être plus flexible sur le changement de l'environnement.

De plus, les utilisateurs doivent s'inscrire auprès de l'administrateur pour fonctionner dans le groupe Normal ou Data Challenge. On peut améliorer la politique pour que la classification de l'utilisateur dans le groupe Normal ou Data Challenge, par rapport le nombre de tâches soumises, se fasse automatiquement. On peut utiliser un seuil: si le nombre de tâches d'utilisateurs est supérieur à ce seuil, l'utilisateur est dans le groupe Data Challenge ; dans

l’autre cas, il est dans le groupe Normal. Ce seuil doit être déterminé au mieux pour la performance de la politique.

7.2.2 Perspective du criblage virtuel au Vietnam

Actuellement le portail de web de criblage virtuel est utilisé pour les besoins des chercheurs à l'INPC. Dans l'avenir, il peut être ouvert aux autres instituts de recherche à Danang ville ou Hochiminh ville où des recherches sur la biodiversité du Vietnam sont également effectuées. On peut également établir un centre de l'informatique qui fournirait des ressources de calcul, des applications et des formations sur l’utilisation de grille/cloud pour la communauté de chercheurs de criblage virtuel au Vietnam.

7.2.3 Perspectives de la grille et du cloud au Vietnam

La technologique du cloud s’est développée dans ces années dernières. La migration technologique de la grille vers le cloud devrait faciliter le déploiement d’une offre de ressources académiques au Vietnam. Grâce aussi à l’amélioration des réseaux pour la recherche et l’éducation au Vietnam (VINAREN), le portail de criblage virtuel devrait trouver un environnement de plus en plus favorable à leur déploiement.

ANNEXE 1. CONSTRUCTION DU PORTAIL DE CRIBLAGE VIRTUEL SUR LA GRILLE

Dans cette partie, nous présentons le travail de Farida Louacheni (louacheni.farida@gmail.com) – une étudiante en Master 2 à l'Institut Francophonie pour l'Informatique qui a construit un portail web de criblage virtuel sur la grille sous l'encadrement de Bui The Quang. Pendant son stage de Master 2, elle a appris le criblage virtuel sur la grille et développé un portail web pour le criblage virtuel en utilisant la grille de calcul pour faciliter la découverte et la recherche de nouveaux médicaments pour les maladies graves et négligées. Son travail nous a permis de proposer une interface conviviale et facile à utiliser pour les utilisateurs non-expérimentés (chimistes, biologistes, médecins…) en informatique et en grille de calcul. Afin de favoriser l'interopérabilité entre le portail web et les services de grille de calcul, nous proposons également une architecture qui permettra une analyse et un traitement fiable des requêtes des utilisateurs finaux.

A1.1 Implémentation

Le but de ce travail est de développer un portail web pour le criblage virtuel, qui permet aux utilisateurs (biologistes, chimistes, bio-informaticiens...) d'envoyer des jobs de docking sur la grille de calcul et de récupérer les résultats à travers ce portail, pour accélérer leur recherche de nouveaux médicaments. Afin de faciliter la tâche aux utilisateurs nous avons proposé une architecture pour le portail que nous allons détailler ci-après. Nous avons déployé plusieurs outils et technologies afin de mener à bien ce projet. Cette partie se concentrera sur l'architecture et la conception du système.

A1.1.1 Architecture du système

Nous avons proposé une architecture simple afin d'offrir aux utilisateurs finaux, qui ne sont pas des experts ni en informatique ni en technologie de la grille, une interface conviviale et facile à utiliser sans qu'ils se soucient de la complexité du portail.

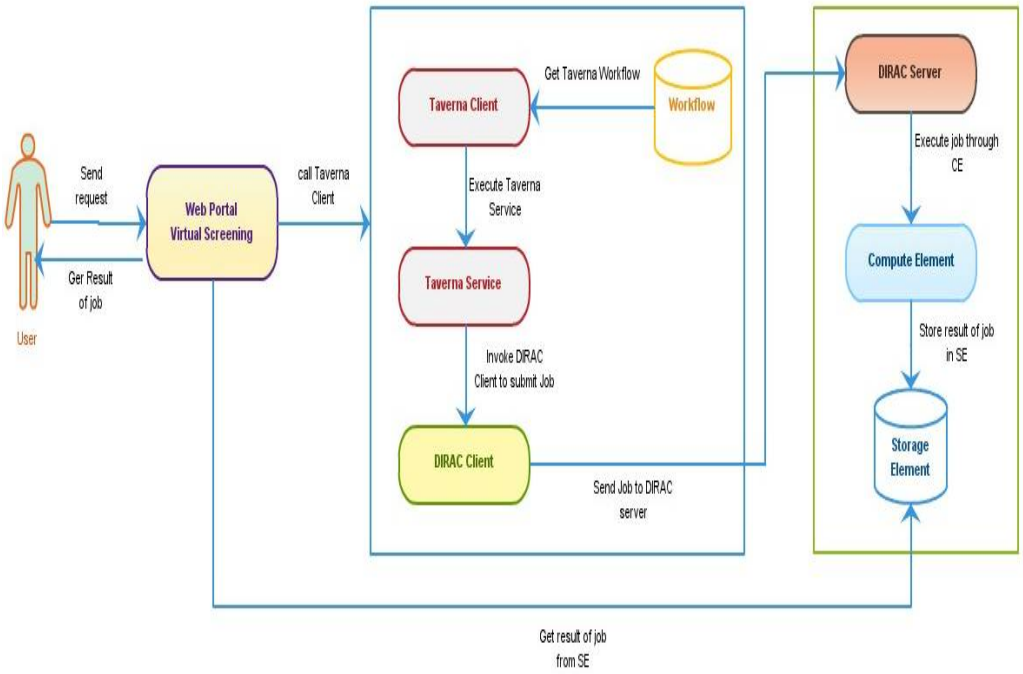


Figure A1-1 - Architecture du système

La figure A1-1 illustre l'architecture que nous avons utilisée pour implémenter le portail web pour le criblage virtuel. L'utilisateur accède au portail web en s'authentifiant avec ses crédentités valides et il envoie sa requête pour effectuer le docking, ensuite, ces paramètres sont affectés comme paramètres d'entrés au Client Taverna. Ce dernier récupère le worflow Taverna, qui contient les services web nécessaires pour le docking (génération et soumission des jobs sur la grille, suivi de l'état du job soumis et récupération du résultat). Après avoir récupéré le workflow, le client Taverna exécute les services pour le docking. L'un des services web est de générer les fichiers "JDL" et de soumettre les jobs sur la grille de calcul en utilisant l'intergiciel DIRAC. Le serveur DIRAC soumet les jobs de docking générés par le service Taverna sur CE de la grille de calcul. Les données stockées sur les SE sont alors transférées sur le nœud de calcul, puis les résultats sont stockés sur un SE de la grille, et répliqués sur d'autres éléments pour réaliser une copie de sauvegarde. A la fin, l'utilisateur peut télécharger les résultats du docking.

A1.1.2 Outils utilisés

Comme illustre l'architecture du système ci-dessus, nous avons utilisé plusieurs outils : AutoDock pour préparer les fichiers nécessaires pour le docking. Pour réaliser ce projet, nous avons utilisé la grille EGI, via la VO Biomed et DIRAC, où un ensemble de scripts avaient déjà été développés, nous avons choisi d'adapter ces scripts sur la grille via DIRAC. L'adaptation a principalement consisté à générer des fichiers JDL, et à utiliser les commandes de DIRAC : "dirac-wms-job-submit" pour soumettre les jobs sur la grille de calcul. Les identifiants des jobs soumis ont été stockés dans un fichier local, qui a ensuite servi à tester le statut des jobs avec la commande "dirac-wms-job-status", et à récupérer les résultats avec la commande "dirac-wms-job-get-output".

La visualisation du workflow s'est faite avec Taverna, qui est un outil pour visualiser le workflow, cet outil est largement utilisé dans le domaine de la bioinformatique, et permet aux utilisateurs d'effectuer des expériences scientifiques. Taverna [78] est un logiciel développé par le consortium myGrid(http://www.mygrid.org.uk/), permettant la réalisation de traitements *in silico* sous la forme de workflows, tout particulièrement dans le domaine de la bioinformatique, cet outil permet l'exécution des expériences sous la forme de workflow. Chaque workflow est constitué par une série de services reliés l'un à l'autre. Taverna est conçu pour combiner des services web distribués et/ou des outils locaux dans des pipelines d'analyse complexes, afin de réaliser la conception et l'exécution des workflows scientifiques. Dans notre projet, nous avons installé Taverna Workbench version 2.5 (http://www.taverna.org.uk/download/workbench/2-5/) pour le mode graphique et Taverna Command Line (2.5) pour le mode ligne de commande (http://www.taverna.org.uk/download/command-line-tool/2-5/), sous le système d'exploitation Linux.

A1.1.3 Développement du portail du web

Nous avons eu recours à plusieurs technologies du web pour le développement du portail. Parmi ces technologies : PHP comme cadre de base et les services web JAX-WS avec JAVA pour interagir avec le système du portail, tout en recevant les requêtes du client. Le portail interagit avec le système via l'intermédiaire du web service. Notre application est réalisée sous Netbeans version 8.0 sous Linux.

Le portail web a été développé en utilisant le framework Yii (Yes It Is), qui est un framework PHP basé sur des composants ultra performants qui a été développé pour créer des applications Web de grande qualité, dont celle d'accélérer le développement d'applications Web. Yii est développé en respectant le modèle MVC (Model-View-Controller).

Les services web

Nous avons créé trois services web en Java sous NetBeans 8.0 avec le serveur GlassFish et Apache Axis2. Chaque service est responsable d'une fonctionnalité pour notre projet. La première fonctionnalité consiste à générer les fichiers nécessaires (script jdl et shell), et à soumettre le job de docking sur la grille de calcul via DIRAC. La deuxième fonctionnalité consiste à suivre l'état du job et à renvoyer son statut, si le statut du job est Failed alors le service va reprogrammer le job et le soumettre à nouveau avec la commande de DIRAC (dirac-wms-file-schedule jobID). La dernière fonctionnalité est la plus importante, elle repose sur la récupération du résultat du job depuis la grille de calcul, si et seulement si le statut du job est à l'état Done. Après avoir exécuté le service web sous Netbeans, un script WSDL (Web Service Description Language) a été généré. Le tableau A1-1 présente en détail les services web que nous avons implémentés, (les paramètres d'entrées, les sorties et une description du rôle de chacun).

Nom de Service	Paramètres	Retour	Description
SubmitJob	- file : String - dpf : String - gpf : String	- Les scripts "jdl" et "shell". - L'identifiant du job (jobID).	- Ce service est évoqué afin de génér les fichiers *jdl* nécessaires pour soumettre un job de docking sur la grille de calcul. Il utilse la commande de dirac : *'dirac-wms-job-submit script.jdl'*. Et la sortie est l'identifiant du job.
getStatus	- jobID : String	- Le statut du job soumet sur la grille de calcul.	- Ce service est responsible sur la récupération du statut du job soumet sur la grille de calcul. Si le statut est "Failed", alors le job est rééchelonner avec la command de dirac : *'dirac-wms-job-reschedule jobID'*
getOutJob	- jobID : String - jobStat : String	- Le chemin du job job soumet.	- Ce service s'occupe de la récupération du résultat du job soumet sur la grille de calcul, en utilisant la commande dirac : *'dirac-wms-job-get-output jobID'*. Il s'agit du chemin où les résulats sont stockés.

Table A1-1 - Description des services web services implémentés

- Le service "submitJob". Ce service récupère les fichiers d'entrées à partir du portail, puis il génère les scripts "jdl" dans un dossier. Après la génération des scripts, les jobs sont soumis sur la grille de calcul à travers l'intergiciel "DIRAC" en utilisant la commande "dirac-wms-job-submit".
- Le service "getStatus". Le role de ce service est de suivre l'état des jobs soumis sur la grille avec la commande "dirac-wms-job-status". Si l'état du job est "Failed", alors le service reprogramme la soumission du job avec la commande de DIRAC "dirac-wms-job-reschedule".
- Le service "getOutJob". Il sert à récupérer le résultat du docking à partir de la grille via la commande DIRAC "dirac-wms-job-get-output" si et seulement si l'état du job est à "Done".

A1.2 Démonstration et résultat

Cette partie se focalisera sur la démonstration du portail web et les résultats du docking. La figure 7-4 présente la fenêtre principale de notre portail. Ce portail a été conçu afin de répondre aux besoins des biologistes, chimistes, bio-informaticiens qui ne sont pas forcément des experts en informatique. Les visiteurs peuvent accéder au portail afin de consulter la liste des ligands, des protéines, des paramètres de grille et des projets qui ont été déjà soumis sur la grille, mais s'ils veulent effectuer une tâche ils doivent s'enregistrer ou s'authentifier en fournissant leur nom et leur mot de passe.

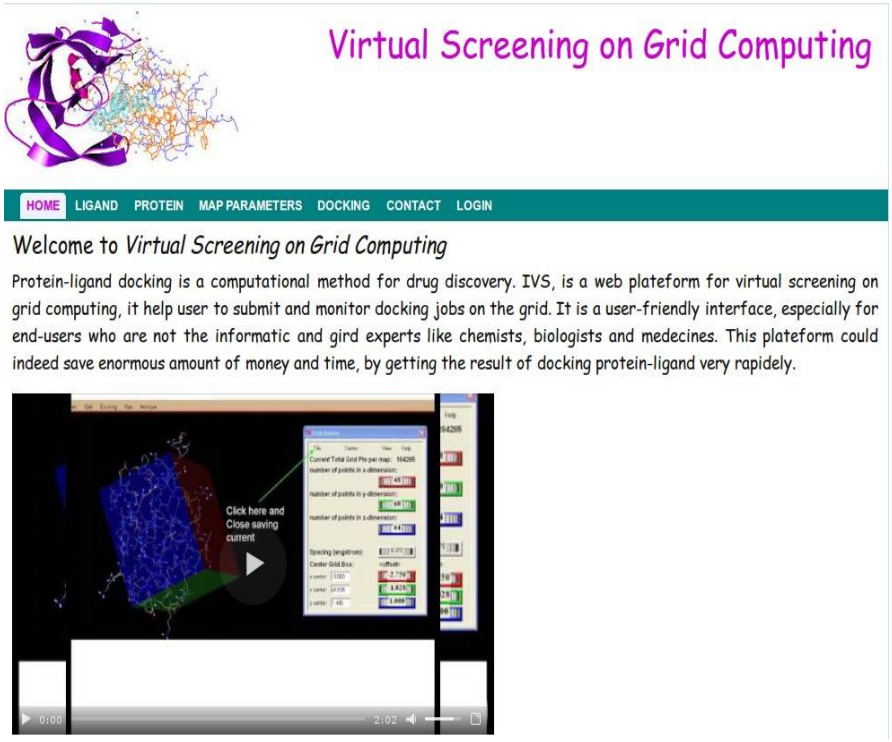


Figure A1-2 : Interface d'accueil pour le portail

L'utilisateur accède au portail par une simple authentification, par le biais de son nom et son mot de passe. La figure ci-dessous montre l'interface pour s'enregistrer auprès du portail afin de profiter de ses services. L'utilisateur entre son nom et son mot de passe.

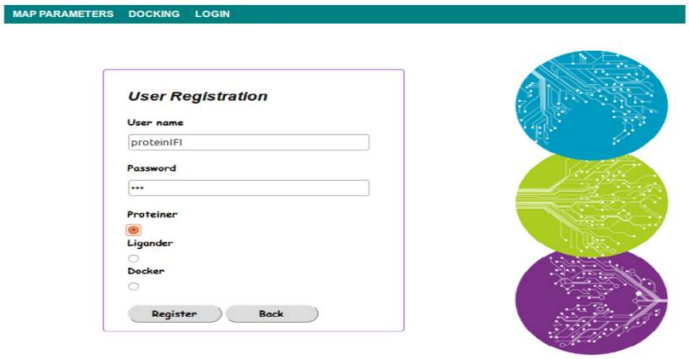


Figure A1-3 : Création d'un nouveau compte

Après la phase d'inscription présentée dans la figure A1-3, l'utilisateur doit s'authentifier pour bien profiter des privilèges qu'un visiteur normal ne possède pas.

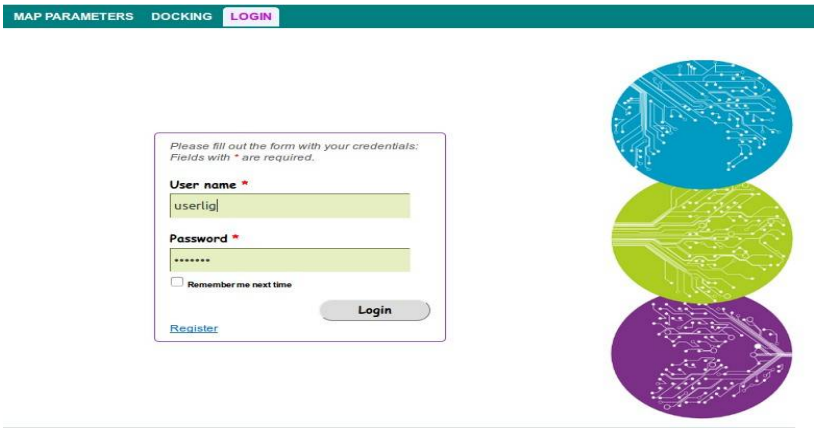


Figure A1-4 : Interface d'authentification

L'administrateur du site possède tous les droits pour la gestion du site. Il gère les utilisateurs, les ligands, les protéines, les paramètres de la grille et les projets. La figure A1-5 donne la liste des utilisateurs.

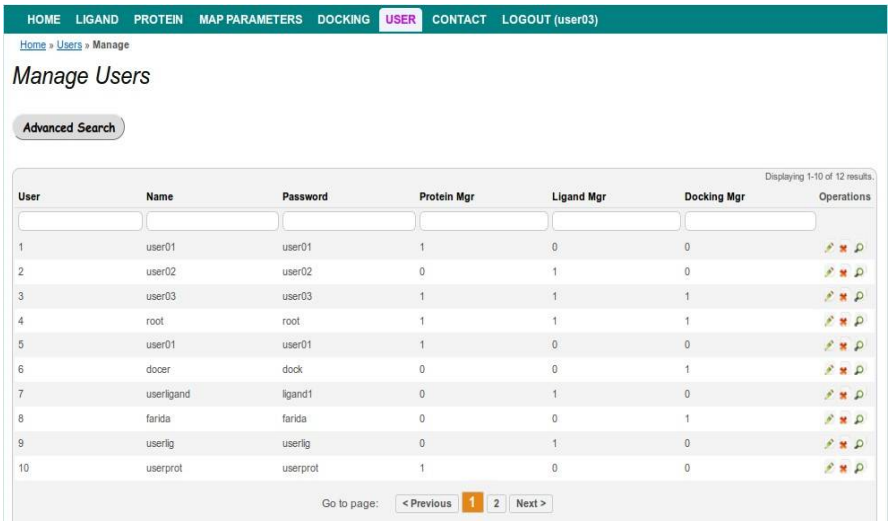


Figure A1-5 : Gestion des utilisateurs

Après s'être enregistré, l'utilisateur peut accéder au portail en fournissant ses crédentités pour accomplir sa tâche. Si l'utilisateur possède un compte, alors il peut ajouter, modifier ou supprimer son ligand et/ou sa protéine. La capture d'écran ci-dessous montre l'ajout d'un nouveau ligand par l'utilisateur de ligand.

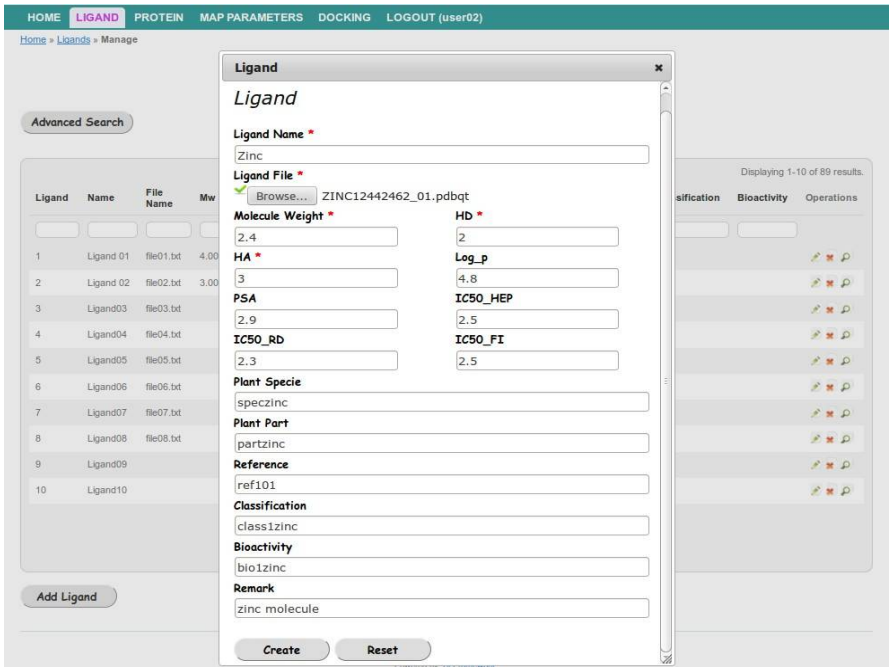


FIGURE A1-6 - Ajout d'un nouveau Ligand

Les autres visiteurs du portail non-authentifié peuvent voir et consulter la liste des ligands disponibles sur le portail et effectuer une recherche sur une molécule de ligand.

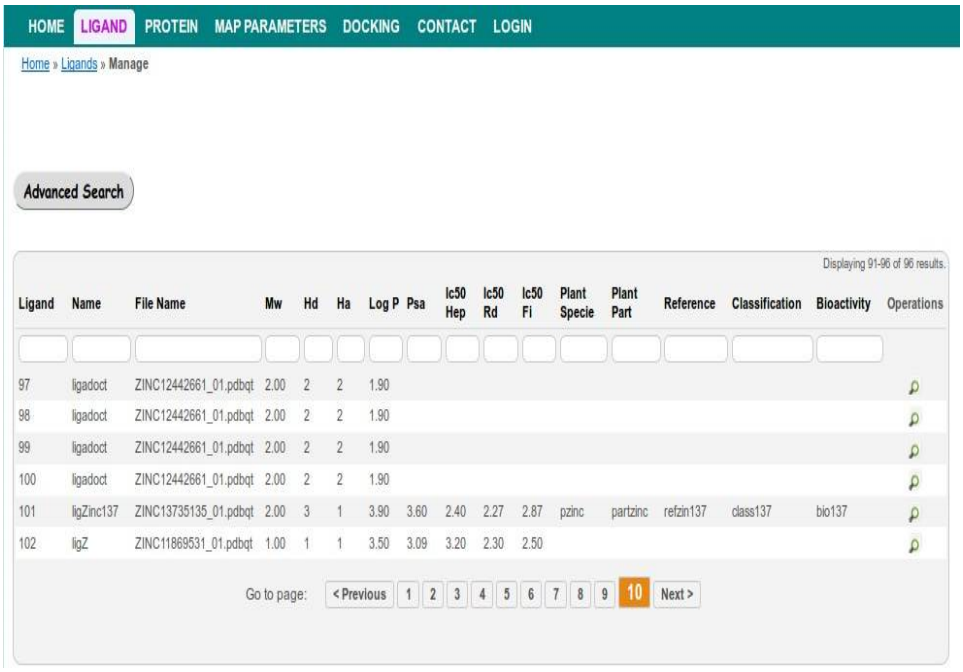


FIGURE A1-7 - Liste des ligands disponibles

L'administrateur de protéine peut gérer la liste de ses molécules de protéines, tout en effectuant les opérations d'ajout, modification et suppression. La figure A1-8 qui montre la fenêtre pour la gestion des protéines.

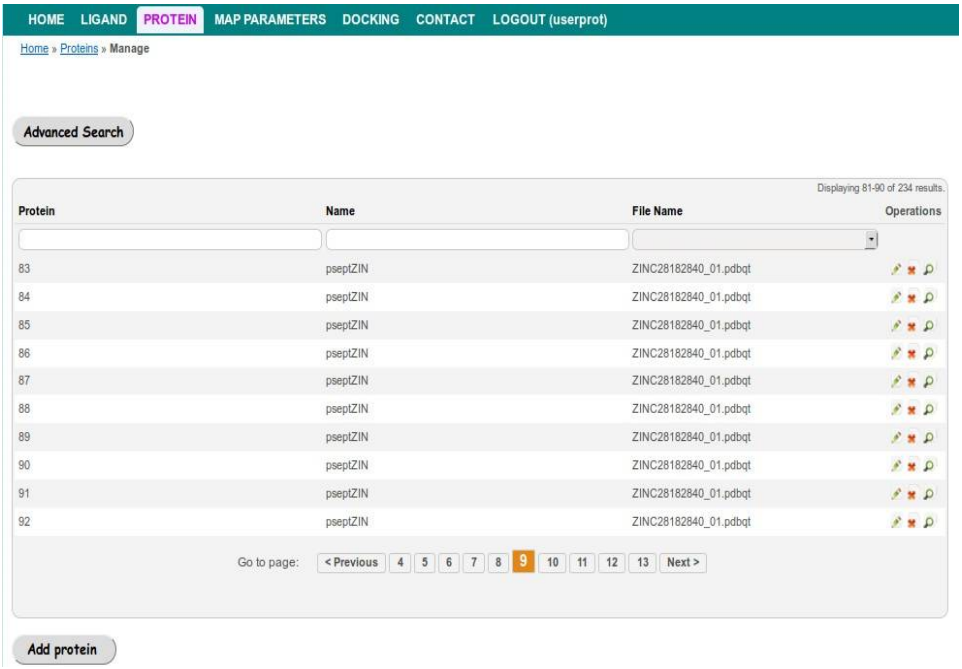


FIGURE A1-8 : Fenêtre de gestion de protéine

La modification d'une protéine nécessite l'authentification de l'administrateur de protéine, comme l'illustre la capture ci-après.

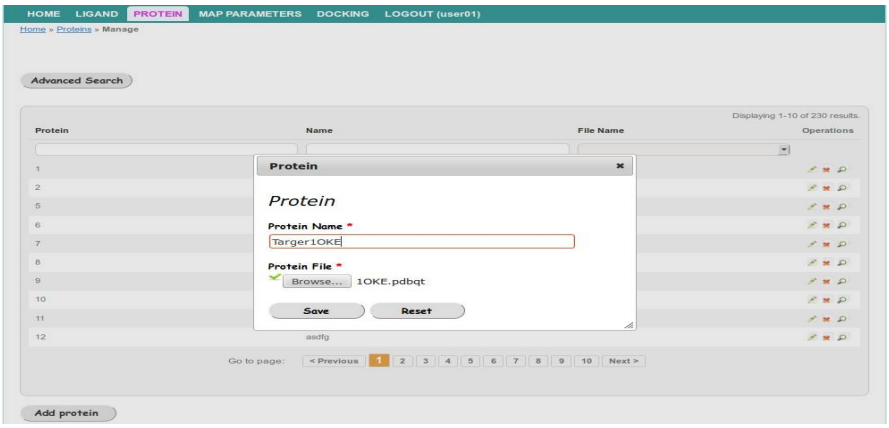


FIGURE A1-9 - Modification d'une protéine

La relation entre la protéine et les paramètres de la grille est une relation 1:n. Tel que, une protéine peut avoir plusieurs paramètres grille, mais un paramètre de grille appartient à une et une seule molécule de protéine.

La capture suivante illustre l'ajout d'un fichier de paramètre de grille.

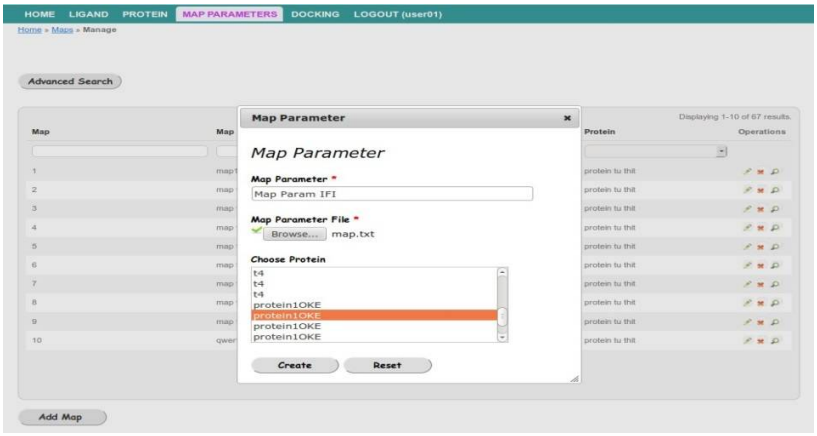


FIGURE A1-10 - Ajout de fichier de grille pour une protéine

On arrive à la partie importante dans ce projet, elle consiste à créer un projet pour le docking et soumettre les jobs sur la grille de calcul. Afin de pouvoir créer un nouveau projet, l'utilisateur doit s'authentifier en fournissant ses crédentités (son nom et son mot de passe), sinon il doit s'enregistrer. Après s'être authentifié, l'utilisateur est capable de créer un nouveau projet, pour cela il doit choisir de la liste un ligand, une protéine, un fichier de paramètre et aussi le fichier de paramètres pour le docking.

Les paramètres ainsi sélectionné sont soit été préparés cet utilisateur soit par d'autres. La capture ci-après illustre la création d'un nouveau projet pour le docking.

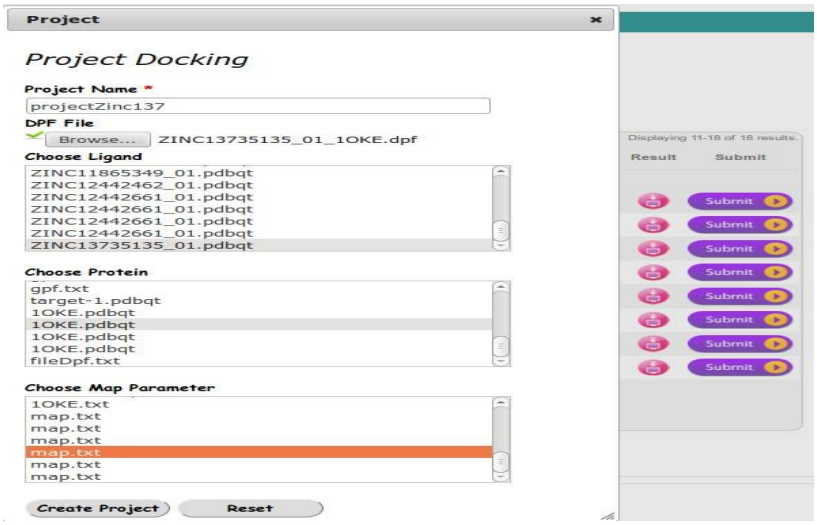


FIGURE A1-11 - Ajout d'un nouveau projet

En retournant à la liste des projets qui ont été créés, on peut vérifier que le projet a été ajouté avec succès.

FIGURE A1-12 - Vérification d'ajout du nouveau projet

Après avoir créé un projet et choisi les fichiers nécessaires pour effectuer le docking. L'utilisateur n'a qu'à appuyer sur le bouton "submit" pour soumettre son job sur la grille. Cette action va permettre de récupérer les fichiers à partir du portail et soumettre le job sur la grille de calcul. Nous allons présenter les résultats que nous avons obtenus lors de soumission des jobs de docking sur la grille, et la récupération des résultats à partir de la grille à travers de ce portail web. Nous nous sommes servis de la base de données protéine-ligand (1OKE pour la protéine, et ZINC pour le ligand) dont le fichier de ligand contient 10256 composants, qui ont déjà été préparés. L'étape de docking moléculaire est réalisée grâce au sous-programme AutoDock4, qui recherche les solutions de docking en fonction des paramètres du fichier "dpf" que l'utilisateur a déjà préparé. Après l'accomplissement du docking, les résultats ont été générés dans des fichiers log avec les extensions glg & dlg :

Le fichier "glg" contient les affinités calculées entre les différents types d'atomes de la protéine et le ligand.

Le fichier "dlg" fournit les coordonnées atomiques des 10 meilleures positions du ligand dans le site de la protéine, leur énergie d'interaction ainsi que les différentes valeurs de l'écart quadratique moyen (Root Mean Square Deviation ou le RMSD).

La figure ci-après montre la soumission de job de docking, tout en sélectionnant le projet "projectZinc137". Dans ce projet nous avons choisi : le fichier de paramètre de docking ZINC13735135_01_1OKE.dpf, le ligand ZINC13735135_01.pdbqt, la protéine 1OKE.pdbqt et le fichier de paramètre de grille map.txt.

FIGURE A1-13 - Soumission de job pour un projet

Le résultat du docking est compressé et stocké dans l'élément de stockage de la grille de calcul, puis on récupère le résultat via le service web afin que l'utilisateur puisse le télécharger. La figure ci-après montre le téléchargement du résultat du job de docking soumis. Le fichier contient les fichiers log de docking et de grille ("dlg" & "glg").

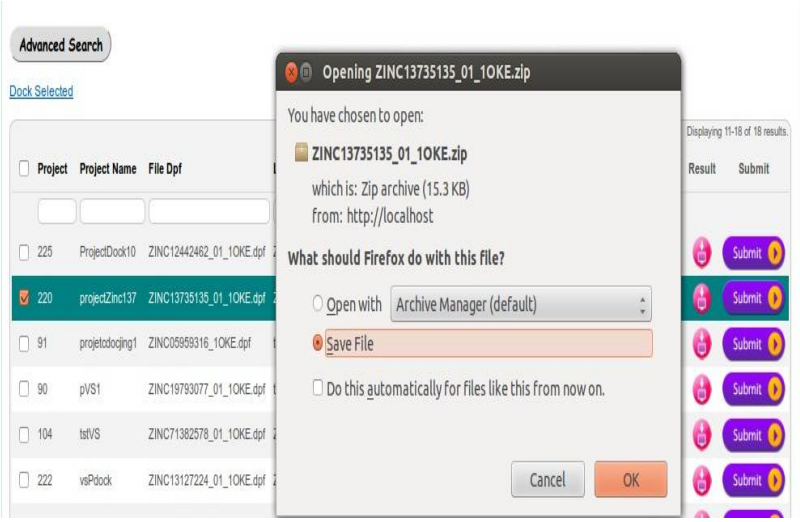


FIGURE A1-14 : Résultat de la soumission de job

Après avoir téléchargé le fichier "zip", nous avons vérifié si l'opération du docking a été effectuée avec succès. La capture ci-dessous illustre le résultat du docking. Comme montre la figure suivante, le docking a été achevé avec succès.

```
AVSFLD: variable 1 file = ZINC11865351_01_1OKE.dlg.pdb filetype = ascii offset = 5 stride = 12
AVSFLD: variable 2 file = ZINC11865351_01_1OKE.dlg.pdb filetype = ascii offset = 6 stride = 12
AVSFLD: variable 3 file = ZINC11865351_01_1OKE.dlg.pdb filetype = ascii offset = 7 stride = 12
AVSFLD: variable 4 file = ZINC11865351_01_1OKE.dlg.pdb filetype = ascii offset = 8 stride = 12
AVSFLD: variable 5 file = ZINC11865351_01_1OKE.dlg.pdb filetype = ascii offset = 9 stride = 12
AVSFLD: variable 6 file = ZINC11865351_01_1OKE.dlg.pdb filetype = ascii offset = 10 stride = 12
AVSFLD: variable 7 file = ZINC11865351_01_1OKE.dlg.pdb filetype = ascii offset = 11 stride = 12
AVSFLD: # end of file

>>> Closing the docking parameter file (DPF)...
This docking finished at:                  5:21 45" a.m., 10/08/2014

/scratch/1618448.ce03.tier2.hep.manchester.ac.uk/CREAM161132108/DIRAC_UTDVTCpilot/19894467/autodock4: Successful Completion on
"wn1205301.tier2.hep.manchester.ac.uk"

Real= 0m 52.10s,  CPU= 0m 51.74s,  System= 0.12s
```

FIGURE 7-15 - Fichier log de docking "dlg"

A1.3 Conclusion

Nous avons proposé dans ce projet un portail web qui répond aux besoins et aux exigences des utilisateurs qui ne sont pas des experts en informatique et qui leur permet d'effectuer le docking *in silico* sans se préoccuper de la complexité du portail en déployant les ressources de la grille de calcul pour soumettre des jobs de docking à travers l'intergiciel DIRAC. Ainsi, nous avons développé et déployé un workflow pour le criblage virtuel sur la grille en utilisant l'outil "Taverna".

BIBLIOGRAPHIE

[1] H. Berman, K. Henrick, and H. Nakamura, “Announcing the worldwide Protein Data Bank,” *Nature Structural & Molecular Biology*, vol. 10, no. 12, p. 980, Dec. 2003.

[2] P. Tollman, P. Guy, J. Altshuler, A. Flanagan, and M. Steiner, “A Revolution in R & D: The impact of genomics,” *The Boston Consulting Group, Boston, MA*, 2011.

[3] P. D. Lyne, “Structure-based virtual screening: an overview,” *Drug Discovery Today*, vol. 7, no. 20, pp. 1047–1055, 2002.

[4] M. Congreve, C. W. Murray, and T. L. Blundell, “Keynote review: Structural biology and drug discovery,” *Drug discovery today*, vol. 10, no. 13, pp. 895–907, 2005.

[5] P. S. Charifson, J. J. Corkery, M. A. Murcko, and W. P. Walters, “Consensus scoring: A method for obtaining improved hit rates from docking databases of three-dimensional structures into proteins,” *Journal of medicinal chemistry*, vol. 42, no. 25, pp. 5100–5109, 1999.

[6] B. Honig and A. Nicholls, “Classical electrostatics in biology and chemistry,” *Science*, vol. 268, no. 5214, pp. 1144–1149, 1995.

[7] M. L. Lamb and W. L. Jorgensen, “Computational approaches to molecular recognition,” *Current opinion in chemical biology*, vol. 1, no. 4, pp. 449–457, 1997.

[8] D. Huang and A. Caflisch, “Efficient evaluation of binding free energy using continuum electrostatics solvation,” *Journal of medicinal chemistry*, vol. 47, no. 23, pp. 5791–5797, 2004.

[9] V. Breton, R. Medina, and J. Montagnat, “DataGrid, prototype of a biomedical grid,” *Methods of Information in Medicine*, vol. 42, no. 2, pp. 143–147, 2003.

[10] E. Laure, A. Edlund, F. Pacini, P. Buncic, M. Barroso, A. Di Meglio, F. Prelz, A. Frohner, O. Mulmo, A. Krenek, and others, “Programming the Grid with gLite,” 2006.

[11] M. Romberg, “The unicore grid infrastructure,” *Scientific Programming*, vol. 10, no. 2, pp. 149–157, 2002.

[12] I. Foster, “Globus toolkit version 4: Software for service-oriented systems,” *Journal of computer science and technology*, vol. 21, no. 4, pp. 513–520, 2006.

[13] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. L. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen, “Advanced Resource Connector middleware for lightweight computational Grids,” *Future Generation computer systems*, vol. 23, no. 2, pp. 219–240, 2007.

[14] F. Hupfeld, T. Cortes, B. Kolbeck, J. Stender, E. Focht, M. Hess, J. Malo, J. Marti, and E. Cesario, “XtreemFS: a case for object-based storage in Grid data management,” in *3rd VLDB Workshop on Data Management in Grids, co-located with VLDB*, 2007, vol. 2007.

[15] J. D. Thompson, D. G. Higgins, and T. J. Gibson, “ClustalW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice,” *Nucleic Acids Research 22*, vol. 22, no. 22, pp. 4673–4680, 1994.

[16] D. J. Lipman and W. R. Pearson, “Rapid and sensitive protein similarity searches,” *Science*, vol. 227, no. 4693, pp. 1435–1441, 1985.

[17] S. Guindon, J.-F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel, “New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0,” *Systematic biology*, vol. 59, no. 3, pp. 307–321, 2010.
[18] B. E. Suzek, H. Huang, P. McGarvey, R. Mazumder, and C. H. Wu, “UniRef: comprehensive and non-redundant UniProt reference clusters,” *Bioinformatics*, vol. 23, no. 10, pp. 1282–1288, 2007.
[19] T.-T. Doan, A. Bernard, A. L. Da-Costa, V. Bloch, T.-H. Le, Y. Legre, L. Maigne, J. Salzemann, D. Sarramia, H.-Q. Nguyen, and others, “Grid-based International Network for Flu Observation (g-INFO),” *Studies in Health technology and informatics*, vol. 159, pp. 215–229, 2010.
[20] P. Mell and T. Grance, “The NIST definition of cloud computing,” *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009.
[21] M. Airaj, C. Cavet, V. Hamar, M. Jouvin, C. Loomis, A. L. Garcia, G. Mathieu, V. Mendez, J. Pansanel, J.-M. Pierson, and others, “Vers une federation de Cloud Academique dans France Grilles,” in *Journ{é}es SUCCES 2013*, 2013.
[22] A. Sartirana, L. Jourdren, P. M. De Freitas, D. Chamont, P. Busson, L. E. Stéphane, and Others, “Eoulsan: analyse du sequencage a haut debit dans le cloud et sur la grille,” in *Journées SUCCES 2013*, 2013.
[23] N. Bard, S. Boin, F. Bothorel, P. Chaumeil, P. Collinet, M. Daydé, B. Depardon, F. Desprez, M. Flé, A. Franc, and others, “E-Biothon: Une plate-forme pour accelerer les recherches en biologie, sante et environnement,” in *Journées SUCCES*, 2013.
[24] B. Depardon, S. Kortas, B. Daix, R. Barate, and others, “SysFera-DS: Un portail d’acces unifie aux ressources des centres de calcul. Mise en application a EDF R&D,” in *Journées scientifiques mésocentres et France Grilles 2012*, 2012.
[25] Vietnamnet, “Cloud computing services becoming more popular in VN,” 2014. [Online]. Available: http://english.vietnamnet.vn/fms/science-it/104073/cloud-computing-services-becoming-more-popular-in-vn.html.
[26] McAfee, “Vietnam a growing cloud computing market,” 2012.
[27] G. M. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, and A. J. Olson, “AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility,” *Journal of computational chemistry*, vol. 30, no. 16, pp. 2785–2791, 2009.
[28] N. Jacq, V. Breton, H.-Y. Chen, L.-Y. Ho, M. Hofmann, V. Kasam, H.-C. Lee, Y. Legré, S. C. Lin, A. Maaß, and others, “Virtual screening on large scale grids,” *Parallel Computing*, vol. 33, no. 4, pp. 289–301, 2007.
[29] L.-M. Birkholtz, O. Bastien, G. Wells, D. Grando, F. Joubert, V. Kasam, M. Zimmermann, P. Ortet, N. Jacq, N. Saïdani, and others, “Integration and mining of malaria molecular, functional and pharmacological data: how far are we from a chemogenomic knowledge space?,” *Malaria journal*, vol. 5, no. 1, p. 110, 2006.
[30] V. Kasam, M. Zimmermann, A. Maaβ, H. Schwichtenberg, A. Wolf, N. Jacq, V. Breton, and M. Hofmann-Apitius, “Design of new plasmepsin inhibitors: a virtual high throughput screening approach on the EGEE grid,” *Journal of chemical information and modeling*, vol. 47, no. 5, pp. 1818–1828, 2007.
[31] G. Degliesposti, V. Kasam, A. Da Costa, H.-K. Kang, N. Kim, D.-W. Kim, V. Breton, D. Kim, and G. Rastelli, “Design and Discovery of Plasmepsin II Inhibitors Using an Automated Workflow on Large-Scale Grids,” *ChemMedChem*, vol. 4, no. 7, pp. 1164–1173, 2009.
[32] H.-C. Lee, J. Salzemann, N. Jacq, H.-Y. Chen, L.-Y. Ho, I. Merelli, L. Milanesi, V. Breton, S. C. Lin, and Y.-T. Wu, “Grid-enabled high-throughput in silico screening

against influenza A neuraminidase.," *IEEE transactions on nanobioscience*, vol. 5, no. 4, pp. 288–95, Dec. 2006.
[33] V. Kasam, J. Salzemann, M. Botha, A. Dacosta, G. Degliesposti, R. Isea, D. Kim, A. Maass, C. Kenyon, G. Rastelli, and others, "WISDOM-II: Screening against multiple targets implicated in malaria using computational grid infrastructures," *Malaria journal*, vol. 8, no. 1, p. 88, 2009.
[34] T. T. Hanh Nguyen, H.-J. Ryu, S.-H. Lee, S. Hwang, V. Breton, J. H. Rhee, and D. Kim, "Virtual screening identification of novel severe acute respiratory syndrome 3C-like protease inhibitors and in vitro confirmation," *Bioorganic & medicinal chemistry letters*, vol. 21, no. 10, pp. 3088–3091, 2011.
[35] T. T. H. Nguyen, H.-J. Ryu, S.-H. Lee, S. Hwang, J. Cha, V. Breton, and D. Kim, "Discovery of novel inhibitors for human intestinal maltase: virtual screening in a WISDOM environment and in vitro evaluation," *Biotechnology letters*, vol. 33, no. 11, pp. 2185–2191, 2011.
[36] H.-Y. Chen, M. Hsiung, H.-C. Lee, E. Yen, S. C. Lin, and Y.-T. Wu, "GVSS: a high throughput drug discovery service of avian flu and dengue fever for EGEE and EUAsiaGrid," *Journal of Grid Computing*, vol. 8, no. 4, pp. 529–541, 2010.
[37] S. Ahn, N. Kim, S. Lee, D. Nam, S. Hwang, B. Koblitz, V. Breton, and S. Han, "Performance analysis and optimization of AMGA for the large-scale virtual screening," *Software: Practice and Experience*, vol. 39, no. 12, pp. 1055–1072, 2009.
[38] E. Yen and S. C. Lin, "Distributed Cloud Development for e-Science," *Proceedings of The International Symposium on Grids and Clouds (ICGC 2012)*, 2012.
[39] H.-K. Wang, "GVSS Portal: A Fully Functional High throughput Screening Web-based Service." [Online]. Available: http://indico3.twgrid.org/indico/contributionDisplay.py?contribId=121&sessionId=31&confId=44.
[40] Anon, "Partnership & Leadership for the nationwide Supercomputing Infrastructure." [Online]. Available: http://www.plsi.or.kr.
[41] S. Rho, S. Kim, S. Kim, S. Kim, J.-S. Kim, and S. Hwang, "HTCaaS: a large-scale high-throughput computing by leveraging grids, supercomputers and cloud," in *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, 2012, pp. 1341–1342.
[42] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, "Job submission description language (jsdl) specification, version 1.0," in *Open Grid Forum, GFD*, 2005, vol. 56.
[43] A. Tsaregorodtsev, M. Bargiotti, N. Brook, A. C. Ramo, G. Castellani, P. Charpentier, C. Cioffi, J. Closier, R. G. Diaz, G. Kuznetsov, and others, "DIRAC: a community grid solution," in *Journal of Physics: Conference Series*, 2008, vol. 119, no. 6, p. 62048.
[44] T. A. Wassenaar, M. Van Dijk, N. Loureiro-Ferreira, G. Van Der Schot, S. J. De Vries, C. Schmitz, J. Van Der Zwan, R. Boelens, A. Giachetti, L. Ferella, and others, "WeNMR: structural biology on the grid," *Journal of Grid Computing*, vol. 10, no. 4, pp. 743–767, 2012.
[45] T. Glatard, C. Lartizien, B. Gibaud, R. da Silva, G. Forestier, F. Cervenansky, M. Alessandrini, H. Benoit-Cattin, O. Bernard, S. Camarasu-Pop, and others, "A virtual imaging platform for multi-modality medical image simulation," *Medical Imaging, IEEE Transactions on*, vol. 32, no. 1, pp. 110–118, 2013.
[46] E. Ostrom, "Tragedy of the Commons," *The New Palgrave Dictionary of Economics*, pp. 3573–3576, 2008.

[47] U. Schwiegelshohn and R. Yahyapour, “Fairness in parallel job scheduling,” *Journal of Scheduling*, vol. 3, no. 5, pp. 297–320, 2000.
[48] F. Pacini, “JDL attributes specification,” *EGEE Document EGEE-JRA1-TEX-555796-JDL-Attributes-v0-1*, vol. 3, 2005.
[49] D. Jackson, Q. Snell, and M. Clement, “Core algorithms of the Maui scheduler,” in *Job Scheduling Strategies for Parallel Processing*, 2001, pp. 87–102.
[50] J. Kay and P. Lauder, “A fair share scheduler,” *Communications of the ACM*, vol. 31, no. 1, pp. 44–55, 1988.
[51] R. L. Graham, “Bounds on multiprocessing timing anomalies,” *SIAM journal on Applied Mathematics*, vol. 17, no. 2, pp. 416–429, 1969.
[52] M. A. Bender, S. Chakrabarti, and S. Muthukrishnan, “Flow and stretch metrics for scheduling continuous job streams,” in *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, 1998, pp. 270–279.
[53] C. Chekuri and S. Khanna, “Approximation schemes for preemptive weighted flow time,” in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002, pp. 297–305.
[54] S. Muthukrishnan, R. Rajaraman, A. Shaheen, and J. E. Gehrke, “Online scheduling to minimize average stretch,” in *Foundations of Computer Science, 1999. 40th Annual Symposium on*, 1999, pp. 433–443.
[55] A. Legrand, A. Su, and F. Vivien, “Minimizing the stretch when scheduling flows of biological requests,” in *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, 2006, pp. 103–112.
[56] E. Medernach, “Allocation de ressources et ordonnancement multi-utilisateurs : une approche basee sur l’equite,” Universite Blaise Pascal - Clermont-Ferrand II, 2011.
[57] L. Eyraud, “Théorie et pratique de l’ordonnacement d'applications sur les systèmes distribués,” l’Institut National Polytechnique de Grenoble, 2006.
[58] G. Schmidt, “Scheduling with limited machine availability,” *European Journal of Operational Research*, vol. 121, no. 1, pp. 1–15, 2000.
[59] E. Caron, “Contribution to the management of large scale platforms: the Diet experience,” Ecole normale supérieure de Lyon - ENS LYON, 2010.
[60] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, “Application-level scheduling on distributed heterogeneous networks,” in *Supercomputing, 1996. Proceedings of the 1996 ACM/IEEE Conference on*, 1996, p. 39.
[61] R. da Silva, T. Glatard, and F. Desprez, “Controlling fairness and task granularity in distributed, online, non-clairvoyant workflow executions,” *Concurrency and Computation: Practice and Experience*, vol. 26, no. 14, pp. 2347–2366, 2014.
[62] T. Glatard, J. Montagnat, D. Lingrand, and X. Pennec, “Flexible and efficient workflow deployment of data-intensive applications on grids with MOTEUR,” *International Journal of High Performance Computing Applications*, vol. 22, no. 3, pp. 347–360, 2008.
[63] J. T. Mościcki, “Distributed analysis environment for HEP and interdisciplinary applications,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 502, no. 2, pp. 426–429, 2003.
[64] V. Kasam, J. Salzemann, M. Botha, A. Dacosta, G. Degliesposti, R. Isea, D. Kim, A. Maass, C. Kenyon, G. Rastelli, M. Hofmann-Apitius, and V. Breton, “WISDOM-II: screening against multiple targets implicated in malaria using computational grid infrastructures.,” *Malaria journal*, vol. 8, p. 88, Jan. 2009.

[65] T. Maeno, “PanDA: distributed production and distributed analysis system for ATLAS,” in *Journal of Physics: Conference Series*, 2008, vol. 119, no. 6, p. 62036.
[66] I. Sfiligoi, “glideinWMS-a generic pilot-based workload management system,” in *Journal of Physics: Conference Series*, 2008, vol. 119, no. 6, p. 62044.
[67] R. G. Diaz, A. C. Ramo, A. C. Agüero, T. Fifield, and M. Sevior, “Belle-Dirac setup for using amazon elastic compute cloud,” *Journal of Grid Computing*, vol. 9, no. 1, pp. 65–79, 2011.
[68] V. M. Muñoz, “How to Run Scientific Applications with DIRAC in Federated Hybrid Clouds,” in *The Seventh International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2013)*, 2013, no. c, pp. 73–78.
[69] T. Fifield, A. Carmona, A. Casajús, R. Graciani, and M. Sevior, “Integration of cloud, grid and local cluster resources with DIRAC,” in *Journal of Physics: Conference Series*, 2011, vol. 331, no. 6, p. 62009.
[70] A. Colin Cameron and F. A. G. Windmeijer, “An R squared measure of goodness of fit for some common nonlinear regression models,” *Journal of Econometrics*, vol. 77, no. 2, pp. 329–342, 1997.
[71] A. Iosup, C. Dumitrescu, D. Epema, H. Liu, and L. Wolters, “An analysis of four long-term Grid traces,” *Technical University of Delft: Delft, Netherlands [Online: http://pds. twi. tudelft. nl/reports/2006/PDS-2006-003/PDS-2006-003. pdf]*, 2006.
[72] E. Medernach, “Workload analysis of a cluster in a grid environment,” in *Job scheduling strategies for parallel processing*, 2005, pp. 36–61.
[73] “The Grid Workloads Archive.” [Online]. Available: http://gwa.ewi.tudelft.nl/.
[74] E. Medernach, “Workload analysis of a cluster in a grid environment,” *Job scheduling strategies for parallel processing*, no. June, 2005.
[75] A. B. Downey, “A parallel workload model and its implications for processor allocation,” *Cluster Computing*, vol. 1, no. 1, pp. 133–145, 1998.
[76] R. Jain, “The art of computer system performance analysis: techniques for experimental design, measurement, simulation and modeling,” *New York: John Willey*, 1991.
[77] D. G. Feitelson, “Packing schemes for gang scheduling,” in *Job Scheduling Strategies for Parallel Processing*, 1996, pp. 89–110.
[78] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, “Taverna: a tool for building and running workflows of services,” *Nucleic acids research*, vol. 34, no. suppl 2, pp. W729–W732, 2006.
[79] Franck Cappello, Samir Djilali, Gilles Fedak, Thomas Herault, Fr_ed_eric Magniette, Vincent N_eri, and Oleg Lodygensky. Computing on Large Scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid. Future Generation Computer Systems, 21(3):417{437, mar 2005.
[80] D. Anderson. Boinc: A system for public-resource computing and storage. In Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Pittsburgh, USA, 2004.
[81] Simon Delamare, Gilles Fedak, Derrick Kondo, and Oleg Lodygensky. SpeQuloS: A QoS Service for BoT Applications Using Best E_ort Distributed Computing Infrastructures. In Proceedings of the 21st ACM International Symposium on High Performance Distributed Computing (HPDC'12), pages 173{186, Delft, The Netherlands, June 2012.

Résumé et mots-clefs en anglais

Abstract: Virtual Screening (VS) is a computational technique used in the drug discovery process to select the most promising candidate drugs for *in vitro* testing from millions of chemical compounds. This method can offer an efficient alternative to reduce the cost of drug discovery and platform. The Natural Products Chemistry Institute of the Academy of Sciences of Vietnam (INPC) collects samples from local biodiversity and determines the 3D structure of single molecules. Their challenge is to set up a virtual screening platform on grid computing for their chemists to process their data. However, as the number of users who might have a wide range of virtual screening applications (in terms of the number of tasks and execution time) increases with limited available computing resources, it becomes crucial to devise an effective scheduling policy that can ensure a certain degree of fairness, user satisfaction and overall system throughput. In this context, the thesis focuses on an effective scheduling policy for the virtual screening workflow where multiple users with varying numbers of tasks are actively sharing a common system infrastructure. We have researched in theory and proposed some candidate policies. With the simulation results and the experimentation results in real system, we proposed the best policy for the fairness between users, which can be applied to INPC virtual screening platform.

Keywords: Virtual Screening, Drug Discovery, Multi-level Scheduling, Fairness, Grid Computing, Cloud Computing.

Résumé et mots-clefs en français

Résumé : l'Institut National des Produits Chimiques de l'Académie des Sciences du Vietnam (INPC) développe depuis plusieurs années une activité autour de la recherche de nouveaux médicaments issus de la biodiversité. Le développement d'un nouveau médicament prend de l'ordre d'une dizaine d'années et passe par plusieurs phases. Dans la phase de découverte, l'activité des composés chimiques sur une cible biologique est mesurée afin de mettre en évidence une action inhibitrice. Le développement d'approches *in silico* pour le criblage virtuel des composés chimiques est une alternative aux approches classiques *in vitro* beaucoup plus coûteuses à mettre en œuvre. L'utilisation de la grille a été identifiée comme une voie économiquement prometteuse pour accompagner la recherche de nouveaux médicaments au Vietnam. En effet, le développement de nouvelles stratégies basées sur l'utilisation de plates-formes de soumission de tâches (DIRAC, HTCaaS) a permis d'améliorer considérablement le taux de succès et le confort des utilisateurs, ouvrant la voie à une démocratisation de la grille.

Dans ce contexte, l'objectif poursuivi dans le cadre de cette thèse est d'étudier dans quelle mesure des plates-formes multidisciplinaires pouvaient répondre aux besoins des chimistes de l'INPC. Le travail s'est concentré sur les modalités d'un partage équitable d'une plate-forme de soumission de tâches sur la grille par une ou plusieurs communautés d'utilisateurs. L'ordonnancement des tâches sur un serveur commun doit permettre que les différents groupes aient une expérience positive et comparable. Sur les infrastructures de grille EGEE et EGI en Europe , on peut distinguer deux grandes catégories d'utilisateurs : les utilisateurs « normaux » qui vont solliciter les ressources pour des tâches requérant typiquement de quelques dizaines à quelques centaines d'heures de calcul, et les « gros » utilisateurs qui vont lancer des grandes productions nécessitant le traitement de plusieurs milliers de tâches pendant des dizaines, voire des centaines de milliers d'heures de calcul. Les stratégies d'ordonnancement déployées aujourd'hui sur les plates-formes comme DIRAC ou HTCaaS ne permettent pas de servir de façon optimale et simultanée ces deux familles d'utilisateurs.

Le manuscrit présente une évaluation par simulation des performances de plusieurs stratégies d'ordonnancement des tâches d'une plate-forme soumettant des jobs pilotes. L'outil SimGrid a permis de simuler l'infrastructure de grille régionale déployée en Auvergne à partir de traces archivées de son utilisation. Après évaluation des performances de plusieurs politiques d'ordonnancement tirées de la littérature, une nouvelle politique a été proposée dans laquelle les utilisateurs normaux et les très gros utilisateurs sont gérés de façon indépendante. Grâce à cette politique, le ralentissement expérimenté par les très gros utilisateurs est réduit significativement sans pénaliser excessivement les utilisateurs normaux. L'étude a été étendue à une fédération de clouds utilisant les mêmes ressources et arrive aux mêmes conclusions.

Les performances des politiques d'ordonnancement ont ensuite été évaluées sur des environnements de production, à savoir l'infrastructure de grille européenne EGI et l'infrastructure nationale de supercalculateurs de la Corée du Sud. Un serveur DIRAC a été adossé aux ressources de l'organisation virtuelle biomédicale d'EGI pour étudier les

ralentissements observés par les utilisateurs de ce serveur. Pareillement, les ralentissements expérimentés par les utilisateurs de la plate-forme HTCaaS au KISTI ont été observés en excellent accord avec les résultats de simulation avec SimGrid.
Ces travaux confirment la faisabilité et l'intérêt d'une plate-forme unique au Vietnam au service des communautés scientifiques consommatrices des ressources académiques de grille et de cloud, notamment pour la recherche de nouveaux médicaments.

Mots-clefs : criblage virtuel, recherche de nouveaux médicaments, ordonnancement, équité, informatique distribué, grilles de calcul, informatique en nuage